opentext™

OpenText™ Information Archive

Installation Guide

Learn how to install the product in demo and production environments, upgrade from a previous version, and disaster recovery strategies. This document is intended for the system administrators responsible for installing OpenText Information Archive.

EARCORE250400-IGD-EN-01

OpenText™ Information Archive
Installation Guide

EARCORE250400-IGD-EN-01
Rev.: 2025-Sept-12

This documentation has been created for OpenText™ Information Archive CE 25.4.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Preface

i Who can install OpenText Information Archive

To install this product, you must have:

- Administrative privileges when installing services for the system's components
- Working knowledge of the operating system that you want to install OpenText Information Archive on
- Working knowledge of a PostgreSQL database deployment
- Working knowledge of web application servers, such as Apache Tomcat

ii Document conventions

This document uses the conventions shown below to aid formatting and readability.

Convention	Description
The `\` newline character in code samples.	Some of the code samples shown in this guide might contain `\` newline characters that allow the code to flow over several lines for readability and formatting purposes. The `\` newline characters are not to be used as part of the code shown, and you must manually remove them prior to using the code.
`<YOUR_PASSWORD>`	A placeholder where you must enter a password.

iii Path and IP address conventions

This document uses the following path and IP address conventions:

`<IA_ROOT>`
: A variable that refers to the directory where you extract the OpenText Information Archive installation package and run the installation script from. By default, this directory is named `infoarchive`, but you can rename it (for example, to include the version number).

  This document uses the `<IA_ROOT>` variable to avoid any confusion when referring to a directory or file path in this documentation.

  For example, when this document refers to the location of the `application.yml` file that is located in the `iaserver` directory, which is located in the `config` directory, this document uses the following path statement:

  ```
  <IA_ROOT>/config/iaserver/application.yml
  ```

Note: In Windows, the name of this directory must only contain alphanumeric characters (letters or numbers) and the characters `.`, `-`, and `_`. Using other characters, including the space character, might cause problems when running the install software.

`<IP_OF_IASERVER>`
A variable that refers to the IP address of IA Server.

`<IP_OF_IAWEBAPP>`
A variable that refers to the IP address of IA Web App.

`<PORT_NUMBER>`
A variable that refers to the port number of a system component.

`<APP_NAME>`
A variable that refers to the directory of an example application (for example, the `Baseball` directory for the Baseball example application.

`<IA_UPGRADE_FROM_ROOT>`
A variable that refers to the directory where you installed the previous version of OpenText Information Archive. `<IA_UPGRADE_FROM_ROOT>` is a separate directory from `<IA_UPGRADE_TO_ROOT>`.

Note: In Windows, the name of this directory must only contain alphanumeric characters (letters or numbers) and the characters `.`, `-`, and `_`. Using other characters, including the space character, might cause problems when running the install software.

`<IA_UPGRADE_TO_ROOT>`
A variable that refers to the directory where you are upgrading OpenText Information Archive to. This directory will contain OpenText Information Archive 23.12. `<IA_UPGRADE_TO_ROOT>` is a separate directory from `<IA_UPGRADE_FROM_ROOT>`.

Note: In Windows, the name of this directory must only contain alphanumeric characters (letters or numbers) and the characters `.`, `-`, and `_`. Using other characters, including the space character, might cause problems when running the install software.

`<IA_SHARED_RESOURCES_DIR>`
A variable that indicates where the shared resources are stored. It strongly recommended that this directory is outside of the `<IA_ROOT>` directory.

Chapter 1

Introduction

OpenText Information Archive is an integrated product suite that is designed for application-agnostic information management and archiving. This product suite preserves, maintains, and controls continuing access to valuable enterprise information assets. With this product, your organization can extend the value of information assets while reducing the costs associated with managing information.

For a complete overview of the product and the concepts behind it, see the *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*.

1.1 System components

OpenText Information Archive Server (IA Server)
This component provides archiving services. It provides a REST API for interaction.

OpenText Information Archive Web Application (IA Web App)
This component provides a web interface for the system. IA Web App provides easy access to applications and their data, searches, and configuration options. Most users access archived data using this web application. IA Web App and Gateway are installed as a single component.

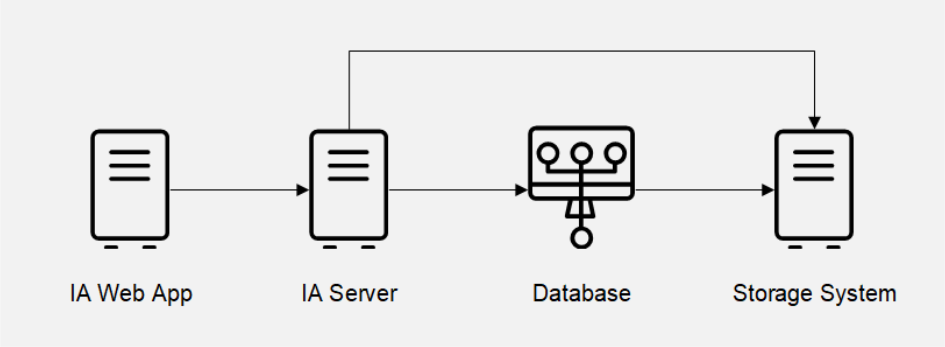
Gateway
Gateway is installed when you install IA Web App. Gateway authenticates users and forwards the web application's REST calls to the IA Server.

Database cluster (PostgreSQL Data Node)
PostgreSQL is used to store the system information. Table applications also require structured data to be stored in PostgreSQL and typically for production environments, different clusters should be used for structured data versus the system data.

Storage systems
Your storage systems store data and metadata for OpenText Information Archive. Supported systems for unstructured data include shared or network-based Windows or UNIX file systems, PowerScale, Elastic Cloud Storage (ECS), and more. There is also a storage API so that you can use a storage system of your choice. Storage systems for information stored in Lucene must be file-system-based. For more information about supported systems, see section 2.2 "Setting up storage for unstructured data" in *OpenText Information Archive - Administration Guide (EARCORE-AGD)* along with its subsections.

IA Server and the database cluster can use the same storage systems to store their data if the storage systems are supported by both components. Throughout this guide, the database cluster and IA Server are often shown in diagrams as sharing the same storage systems, for the sake of simplicity. Storage systems are

also shown in diagrams as part of the overall solution, but you supply the storage systems.



REST API
This API allows you to interact programmatically with IA Server. For more information, see *OpenText Information Archive REST API Developer's Guide* on support.opentext.com (https://support.opentext.com/).

OpenText Information Archive Shell (IA Shell)
This component is a command-line tool for performing administrative tasks, ingesting data, and managing or querying objects. It makes calls to the REST API to perform these functions. You can also use the IA Shell with scripts. Users typically interact with the system through IA Web App or the IA Shell, depending on their preferences and the operations that they want to perform (not every option is available in both). For more information, see the *OpenText Information Archive - IA Shell Guide (EARCORE-ARE)*.

1.2 System directories

The system's installation package consists of `ZIP` files that include the files and directories required to install and run OpenText Information Archive.

- `infoarchive`: This is the root directory in the ZIP file.

The following directories are in the root directory:

- `bin`: Contains key scripts and utilities, including the following:
  - Startup scripts for various components.
  - Utilities, such as the IA Shell and password encryption tools.
- `config-templates`: This directory contains configuration file templates used by the setup/install software to generate the configuration files that will be used by the various components. You can treat this directory like a form of documentation. It contains all the possible parameters, with default values, for the files that are in the `config` directory, and includes comments about the parameters. You can use the files in the `config-templates` directory to understand which parameters are supported in the files in the `config` directory.

Note: Do not change the files in the `config-templates` directory. Doing so can break the generation of config files using setup.

- `config`: This directory contains generated configuration files. These files are generated by the install software, and it is recommended to change them through the use of the install software as much as possible. However, there are some types of configuration changes that the install software does not cover. These changes require manual modifications to files in this directory.
- `data`: This directory is used by demo installations as a default location for filesystem-based stores. For production environments, this directory must not be used and instead a directory should be used which is outside of the distribution (`<IA_ROOT>`) directory structure. In case of more than a single IA Server node, this directory must be accessible to all IA Server nodes and thus needs to be either network-mapped or shared explicitly.

  It is also separate from the shared resources directory, which is used for configuration files such as keystores, *etc.* and do not constitute data being managed by OpenText Information Archive.
- `extensions`: Used for OpenText Information Archive extensions and includes some example extensions.
- `first-time-setup`: Contains several key items for setting up OpenText Information Archive:
  - The `PSQL_xxxxxx_IA.txz` file, which is the PostgreSQL installation distribution. There are two versions, one for Linux and one for Windows.
  - The first-time setup applications and their installation and upgrade scripts: `install.bat` (Windows) and `install` (Linux).
  - System files for Windows and Linux related to the installation of services.
- `lib`: Contains program libraries for OpenText Information Archive. It also contains configurations for different store types for declarative configurations.
- `license`: Contains license agreements, including licenses for third-party components, and a third-party component listing.

1.2.1 infoarchive-support jar

The `infoarchive-support` jar does not need to be installed but contains documentation and examples to help better understand the product.

The following can be found in the `infoarchive-support` directory:

- `doc`: Contains JavaDocs with information on creating custom handlers and custom stores, and REST documentation. Includes a `ZIP` file containing HTML pages that describe the REST resources.
- `examples`: Contains example OpenText Information Archive applications and their data, and some supporting examples of Spring Boot applications using REST.

1.3 Storage considerations

The system supports several vendor-neutral network-attached storage (NAS) solutions, and it provides you with an extensible storage API framework that you can use to integrate OpenText Information Archive with a custom storage solution. For more information, see section 2.1 “Setting up storage for structured data” in *OpenText Information Archive - Administration Guide (EARCORE-AGD)* and section 2.2 “Setting up storage for unstructured data” in *OpenText Information Archive - Administration Guide (EARCORE-AGD)*.

1.4 Specifying parameters in configuration files

When you specify a parameter in a configuration file in YAML format, be careful not to include any extra spaces at the end of the value. An extra space might cause the value to be interpreted incorrectly. Also, be careful to keep a single space between a colon (:) at the end of a parameter name and the value for the parameter (for example, `key: value`). If the space between the parameter name and its value is missing, the configuration file might also be parsed incorrectly.

Chapter 2

Requirements and supported configurations

2.1 Environment and system requirements

Before you perform an installation, refer to the latest version of *OpenText Information Archive Release Notes* on support.opentext.com (https://support.opentext.com/) for certified environment and system requirements, as well as known issues and workarounds.

Important

OpenText Information Archive patches can be installed **without** having the product or previous patches already installed. This means you could do a clean install with any patch version of the current version.

If you want to install components on a Linux computer, make sure that the computer consistently has high entropy. There are many ways to do this, including running the haveged daemon or the `rng-tools` package. Without sufficient entropy, performance may be greatly reduced, and processes can start to hang while waiting for sufficient entropy to build up while performing cryptographic operations.

2.2 Demo configuration

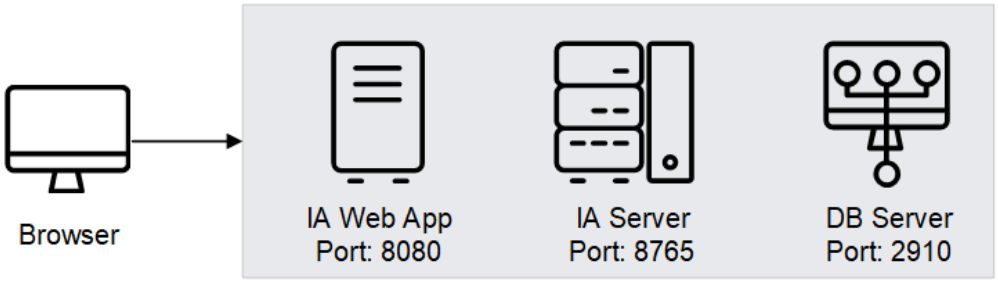
You can quickly set up a demo configuration of OpenText Information Archive so that you can test its features, set up a proof of concept, give a short demonstration, or set up a basic development environment. In a demo configuration, you install all the components on one computer and run IA Web App as a standalone Spring Boot application on the same computer. The demo configuration includes some example applications, data, and user accounts.

Caution

Do not use a demo configuration in a production environment or convert it for use in a production environment.

For more information, see Quickly installing a demo configuration.

The following illustrates a demo configuration, with all components on the same computer:



Related information

- Production configuration

2.3 Production configuration

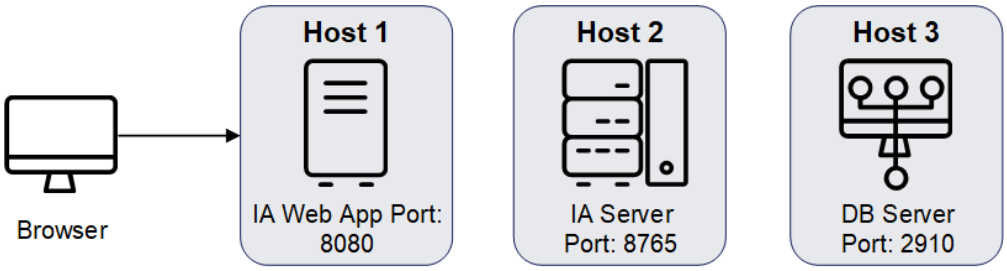
For a production configuration, you should distribute components over separate computers, including running IA Web App on a Tomcat server. As part of this, you should make sure that for any file-based systems, the distributed components can access this information. This in particular applies to the file system roots used as the back end storage system for file system-based content stores used for Lucene and export type stores. This file system root(s) will need to be either network mapped or explicitly shared so it is read-write accessible to all IA Server nodes to avoid unpredictable problems during a large variety of operations.

You must configure at least one data node for the system. It is recommended that you configure at least one data node for table applications. If you have no table applications other than the Reports application, you can share the data node for the system. How you choose to distribute the components depends on your operational and security requirements, as well as any concerns about network latency.

A production configuration, due to its distributed setup, has significant advantages over a demo configuration, such as high availability, scalability, increased performance, and recoverability. It is difficult to convert a demo configuration into a production configuration without significant downtime and data migration, so you should plan ahead for a production configuration instead.

The following diagram illustrates two example configurations using default ports. In the demo configuration, all the components are installed on the same computer. In the production configuration, each component is installed on a separate computer.

The following illustrates a production configuration, with all components on different computers:



Related information

- Demo configuration

2.4 OpenText Information Archive applications

Applications provide access to archived data from source applications. There is a typically a separate application for each source application (for example, patents) or for each logical collection of data that you are drawing from multiple source applications (for example, sales histories).

An application has its own storage settings, and it must have adequate storage to work properly. Each application acts as a self-contained unit.

Users typically access an application with IA Web App, which allows them to search data.

2.4.1 Types of applications

Different types of applications are built using different methods or perform different functions. All applications make REST API calls to interact with the system.

Example
OpenText Information Archive is bundled with many example applications in declarative configuration format.

First-time setup
OpenText Information Archive includes four applications that you must install to enable key functionality used throughout OpenText Information Archive: System, Reports, Audit, and Tenant. These are also known as core applications.

IA Web App
You can use IA Web App to create an application. While this method automates some steps of the creation process, there are other steps that you must perform manually, using a declarative configuration.

2.4.2 Example and first-time setup applications

OpenText Information Archive includes example applications, which can help you understand how to create applications and work with data. You can also make changes to the example applications and use them to archive data from your own source applications. The example applications use property files that can control the mode to use for SIP-based applications.

Different examples illustrate different use cases, such as application decommissioning versus live archiving, or table archiving versus compound record archiving.

Example applications are stored in the `<IA_ROOT>/examples/applications` directory and are only installed if the `infoarchive-support zip` is unzipped.

First-time setup applications provide key functionality to OpenText Information Archive. For example, the System application stores diagnostic logs for job instances and certain order items, as well as runtime processing data for several operations.

Audit is used to archive audits into and make them available for searching. Reports is used to provide searches over previously generated statistics through the Refresh Metrics job.

Tenant is not an application but contains several components that are not application-specific and can be used by the system or multiple applications, such as export configurations.

First-time setup applications are stored in the `<IA_ROOT>/first-time-setup/applications` directory.

The following illustrates the differences between example and first-time setup applications:

Type of IA app	Application name	Type of use case
Example	Baseball	Application decommissioning
	OrderManagement	Table archiving
	Order_Management_Multilanguage	
	Patent	
	Tickets	
	PhoneCalls	Live archiving
	PhoneCallsGranular	SIP archiving
	Certificates	
	Invoices	
	Trades	

Type of IA app	Application name	Type of use case
	CertificatesAndTrades	
	CreditTresor	
	SEPA	
	SAPConnector	Archiving using connectors for the ETL process
	SharePoint	
First-time setup	System	Storing and persisting diagnostics logs for job instances and certain order items. Contains properties for default stores and storage, including default stores for search results that are used for export functionality. The configuration is hidden, and this application is not visible in the list of installed applications.
	Audit	Storing and searching the audit event types in OpenText Information Archive
	Reports	Storing metrics, such as reports for upcoming and overdue dispositions
	Tenant	Installing the default, tenant–scoped search transformations, a custom presentation view, a set of export pipelines, and sample branding configuration files. Technically, this is more a configuration that can be shared across applications rather than an application.

Type of IA app	Application name	Type of use case
Example	Baseball	Application decommissioning Table archiving
	OrderManagement	
	Order_Management_Multilanguage	
	Patent	
	Tickets	
	PhoneCalls	Live archiving SIP archiving
	PhoneCallsGranular	
	Certificates	

Type of IA app	Application name	Type of use case
	HR	
	Invoices	
	Trades	
	CertificatesAndTrades	
	CreditTresor	
	SEPA	
	Documentum	
SAPConnector	Archiving using connectors for the ETL process	
SharePoint		
First-time setup	System	Storing and persisting diagnostics logs for job instances and certain order items. Contains properties for default stores and storage, including default stores for search results that are used for export functionality. The configuration is hidden, and this application is not visible in the list of installed applications.
	Audit	Storing and searching the audit event types in OpenText Information Archive
	Reports	Storing metrics, such as reports for upcoming and overdue dispositions
	Tenant	Installing the default, tenant–scoped search transformations, a custom presentation view, a set of export pipelines, and sample branding configuration files. Technically, this is more a configuration that can be shared across applications rather than an application.

For more information about how each of the example applications demonstrates specific features of the product, see How example applications demonstrate product features.

2.5 Connectors for the ETL process

Connectors for the ETL process transform data from its source format in the source application into XML for an OpenText Information Archive application to ingest. You can use the connectors that are described below, you can use partner tools, or you can create your own connectors.

These are example applications that were created using the following OpenText Information Archive connectors:

- OpenText Information Archive SharePoint Connector
- OpenText Information Archive for SAP Connector

As the Documentum Connector has been removed, the functionality has been merged and improved inside Documentum Content Server since 20.2. It provides a better integration and can be used on previous versions based on supported DFC versions.

OpenText Information Archive connector examples can help you understand how to use applications with the OpenText Information Archive connectors to archive data from SharePoint, and SAP. These example applications are not the connectors themselves but include SIPs that were created using the connectors.

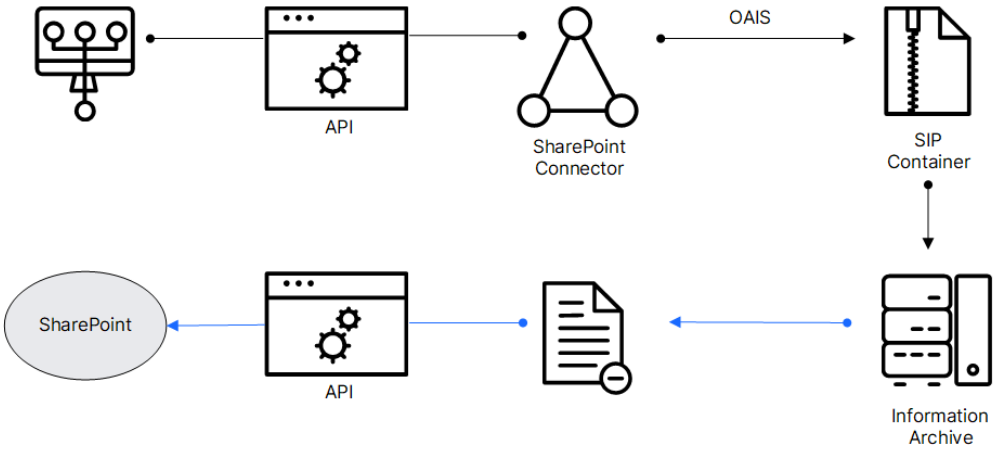
These connector examples have also been included with several previous versions of the product, so if you have used a previous version of OpenText Information Archive, they can help you understand how the product's functionality has changed.

2.5.1 OpenText Information Archive SharePoint Connector

OpenText Information Archive SharePoint Connector is a command-line utility for data extraction and transformation that works with Microsoft SharePoint. You can use it to extract list items from SharePoint sites and generate SIPs for OpenText Information Archive to ingest.

OpenText Information Archive SharePoint Connector extracts list items from SharePoint sites, saves the attributes of extracted items in the SIP's PDI file, and creates the SIP descriptor based on the settings in the configuration file. For more information, see section 2 "OpenText Information Archive SharePoint connector" in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.

The following diagram illustrates how SharePoint Connector works with SharePoint and OpenText Information Archive.



You can use SharePoint Connector with SharePoint 2010, 2013, and SharePoint Online as a part of an Office 365 suite.

2.5.2 OpenText Information Archive for SAP Connector

The OpenText Information Archive for SAP Connector provides a technology bridge between an SAP system (R/3 ERP or ECC) and OpenText Information Archive, using the SAP HTTP ArchiveLink interface. You can use it to archive SAP data, reports, and documents using the ArchiveLink certified interfaces in OpenText Information Archive. For more information, see section 1 “OpenText Information Archive for SAP connector” in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.

The OpenText Information Archive for SAP Connector enables you to access and display documents that are stored in OpenText Information Archive from within a variety of SAP modules. It also enables you to use the user interface to search for and view documents that were archived from SAP into the OpenText Information Archive repository.

Chapter 3

Quickly installing a demo configuration

A demo configuration is a quick way to install all the OpenText Information Archive components on a single computer. A demo configuration is not meant to be used in a production environment or converted for use in a production environment. It includes the following:

- Example applications and data
- Auto-generated passwords for OpenText Information Archive components
- No encrypted passwords, TLS/SSL, LDAP, or Microsoft Active Directory
- Database connections using the superuser account
- Example user accounts for accessing OpenText Information Archive

Caution

- The instructions below are for setting up a demo configuration, not a production configuration. A demo configuration has little resemblance to a production configuration, where high availability, performance, recoverability, and so on are considerations. It is difficult to transition from a demo configuration to a production configuration without significant downtime and moving data around. To plan for a production configuration, see Planning a production configuration and Installing a production configuration.
- Do not use the example user accounts in a production environment. Even enabling these accounts significantly undermines any security measures implemented and easily allows attackers to use one of the example user accounts to gain unauthorized access to your OpenText Information Archive system and the assets that it contains. These out-of-the-box accounts are meant for demo and limited configuration purposes only, and the default password for each account is `password`. For more information about example user accounts, see Working with example user accounts.

When you install OpenText Information Archive, you can choose whether to install components as system services. If you choose not to, you can start and stop the components manually using a command prompt, which also lets you monitor any messages, allowing you to troubleshoot any startup problems.

3.1 Installing a demo configuration

After you acquire the latest OpenText Information Archive distribution `ZIP` files, follow these steps to install a demo configuration.

Make sure to unzip `infoarchive.zip`.

Before you begin:

- Refer to the latest version of *OpenText Information Archive Release Notes* on support.opentext.com (https://support.opentext.com/) for certified environment and system requirements, including the required version of JDK.
- Make sure to download the latest patch so that you have all the latest software fixes. Each patch contains the full binary of the product and can be installed without having the product or previous patch already installed.

To install a demo configuration:

1. Open a command prompt, go to the `<IA_ROOT>` directory. Type one of the following install commands:
   - Windows: `install.bat`
   - Linux: `install`
2. The following prompt appears:

   ```
   Install a demo configuration? (Y/n)
   ```

   Press **Y** and press **ENTER**.

3.2 Starting the components with a command prompt

Before you can use OpenText Information Archive, you must start the components. When you start the components using a command prompt, you can view status and error messages for the components. For this reason, starting the components with a command prompt is useful for making sure that the components are working properly. When you close the command prompts, the OpenText Information Archive components stop running.

If you do not configure the components to run as system services, you must start the components with a command prompt.

If you encrypt component passwords and secret tokens, you can only start the components with a command prompt, unless you configure the components to start without password input. For more information, see section 7 “Encrypting component passwords and secret tokens in configuration files” in *OpenText Information Archive - Encryption Guide (EARCORE-AGE)*.

To start the components on a Windows computer:

1. Open a command prompt, go to the `<IA_ROOT>\psql` directory, and then type the following command:

   ```
   bin\pg_ctl.exe -Dconf start
   ```

   Leave this command prompt open. If you do a **CTRL+BREAK** in this window, it will stop the database.

2. Open a new command prompt, go to the `<IA_ROOT>` directory, and then type the following command:

   ```
   bin\iaserver
   ```

   Leave this command prompt open.

3. Open a new command prompt, go to the `<IA_ROOT>` directory, and then type the following command:

   ```
   bin\iawebapp
   ```

   Leave this command prompt open.

To start the components on a Linux computer:

1. Make sure that there is enough entropy available for cryptographic applications (for example, run the Haveged daemon or `rng-tools`).

2. Open the Linux Command Shell, and type the following commands:

   a. `<IA_ROOT>/psql/bin/pg_ctl -D <IA_ROOT>/psql/conf start &`

   b. `<IA_ROOT>/bin/iaserver &`

   c. `<IA_ROOT>/bin/iawebapp &`

3.3 Installing applications for a demo configuration

Before you use a demo configuration, you must install the first-time setup applications, and you might also want to install some example applications.

Note: It is not recommended to use the download IA Shell to install any of the sample applications. The sample applications can only be installed from a machine where the setup program was run and the option to configure applications was specified. Unless the `default.properties` file was generated, attempting to install these applications will result in multiple failures.

You must install the first-time setup applications (System, Audit, Reports, and Tenant) because they enable key functionality used throughout OpenText Information Archive.

The Baseball and PhoneCalls applications are helpful examples for understanding how to create OpenText Information Archive applications and work with data. These applications have also been included with several previous versions of

OpenText Information Archive, so if you have used a previous version, they can help you understand how the product's functionality has changed.

- Baseball is a table application that serves as an example of archiving structured data.
- PhoneCalls is a SIP application that serves as an example of archiving compound records, with structured data and unstructured content.
- Other example applications are also available.

For more information about all the applications that are included with OpenText Information Archive, see Connectors for the ETL process.

The installation process described below has been simplified for demo configurations and assumes that there will be enough disk space for your data. If you want to configure any parameters (for example, if disk space is a concern and you want to configure storage parameters).

If you are performing a demonstration and you chose not to enable the example user accounts, then you must set up a user account that has access to all roles and use that account to access the application.

After you install the Audit and Reports applications, if you want to search the audits and run reports, then in IA Web App, run the Refresh Metrics job and then the Archive Audits job. These jobs populate the applications with the necessary data. For more information about running a job, see section 3.6 "Running and viewing metrics with the Administration dashboard" in *OpenText Information Archive - Administration Guide (EARCORE-AGD)* and section 3.7.29.5 "Running the job ad hoc" in *OpenText Information Archive - Administration Guide (EARCORE-AGD)*.

To install the first-time setup applications:

1. Open a command prompt, go to the `<IA_ROOT>/first-time-setup/applications` directory, and then type the following command:

   - Windows: `install.bat`
   - Linux: `./install`

   The following two steps are only required if `userName` and `userPassword` credentials were not supplied in the `config/iahsell/application.yml` file.

2. The following prompt appears:

   ```
   Please, enter user name:
   ```

   Type the user name that you want to use to install the applications.

3. The following prompt appears:

   ```
   Please, enter password for user <USER_NAME>:
   ```

   Type the password for the user account.

To install an example application:

1. On a computer that hosts an OpenText Information Archive component, open a command prompt, and go to the `<IA_ROOT>/examples/applications/<APP_NAME>` directory.
2. Type one of the following commands:
   - Windows: `install.bat`
   - Linux: `./install`

3.4 Verifying the demo configuration

You can log into a demo configuration using any of the example accounts, but you might want to log on for the first time as `connie@iacustomer.com`, an account with Developer privileges.

To verify the demo configuration:

1. Open a supported browser and go to `<IP_OF_IAWEBAPP>:<PORT_NUMBER>` (for example, `192.168.0.13:8080`).
2. Log in using a user name and password.
3. Install at least one of the example applications.
4. To verify that OpenText Information Archive ingested data from an example OpenText Information Archive application, perform a search.

3.5 Running components as system services

In a production environment, in most cases, you should consider installing OpenText Information Archive components as system services so that the operating system automatically tries to restart a service if it stops unexpectedly. Alternatively, you can start and stop components manually using a command prompt. If you opt for this approach, you can automatically start and stop them using your startup scripts called by your enterprise monitoring and scheduling tool.

The scripts for installing services for OpenText Information Archive components do not support installing multiple instances of the same component on the same computer (for example, multiple instances of IA Server on the same computer). If you want to run multiple instances, you should use your enterprise monitoring and scheduling tool to automatically start and stop the components.

In general, it is a good idea to at least verify the configuration by starting all components manually first, and only rely on services once it has been verified that they all work together using the current configuration without problems.

On Windows, the key prerequisite when installing components as system services is to verify that you have administrative privileges. Also, if the IA Server service is set up to run as a local system user, there might be a permission issue for a Windows

share when working with UNC paths for filesystem storage. You must run the IA Server service with the service account.

On Linux, the key prerequisite when installing components as system services is to verify that you have root privileges. However, after installation, you might not want to run the system services with a user account that has root privileges. In this case, you must make sure that the permissions for the configuration files are readable by the user account that you want to run the system services (also known as the service account).

To make sure that the permissions for the configuration files are readable by the services account, you might want to do one of the following:

- Change the owner of all files under `<IA_ROOT>` to the service account
- Change the file permissions of all files under `<IA_ROOT>` to be readable by the service account
- Change the file permissions of only the relevant files under `<IA_ROOT>` to be readable by the service account

For more information about the files and directories contained in `<IA_ROOT>`, see OpenText Information Archive directories.

3.6 OpenText Information Archive setup: demo vs. production

For more information about planning a production configuration, see the *OpenText Information Archive Setup: Demo vs. Production* whitepaper in the Documentum Champion Toolkit, available on support.opentext.com (https://support.opentext.com/).

Chapter 4

Security

4.1 Component passwords and client secrets

The system uses several passwords and client secrets for its components to communicate. For example:

Master component	Referring component	Passwords and client secrets
PostgreSQL	IA Server	• Password for each data node • Password for each database
IA Web App (Gateway)	JDBC	Secret client key for obtaining a token for JDBC from the Gateway.
IA Web App (Gateway)	IA Shell	Secret client key for obtaining a token for IA Shell from the Gateway.
IA Web App (Gateway)	IA Server	Secret client key for communication between IA Server and the Gateway.

In a demo configuration, these passwords and client secrets are automatically generated. In a production configuration, you can choose to auto-generate them, or to manually specify your own values, and you can have them encrypted automatically according to settings that you provide. It is also possible for the passwords to be stored in a Vault.

4.2 Authenticating users

Important

OpenText Information Archive has deprecated support for LDAP and Active Directory (AD). Integration of these communication protocol system will be removed in a future release. Use of LDAP and AD can be done using OTDS. It is recommend to use OTDS instead as soon as possible.

Caution

Do not leave the example user accounts enabled in a production environment. Attackers can use one of the example user accounts to gain unauthorized access to your system and the assets that it contains. These out-of-the-box accounts are meant for demo and limited configuration

purposes only, and the default password for each account is `password`. For more information about example user accounts, see Working with example user accounts.

OTDS integrates OpenText enterprise information management (EIM) products and solutions with a company's enterprise directory infrastructure (based on AD and/or LDAP). You may decide to enable single sign on (SSO) across all components out-of-the-box.

The solution facilitates seamless integration in two areas. In the first one, OpenText EIM products and applications are tied to a company's infrastructure concerning identity management. The second area connects OpenText EIM components together to have the same logical view on users and groups, so each user may need to log in only once (maybe just to the desktop) to work with an application that spans multiple systems (for example, ECM and BPM, or SAP and ECM).

OTDS is a separate download that you install separately from OpenText Information Archive, but it does not require an additional license.

For more information, see section 7.1.1 "Authenticating users – configuring OpenText Directory Services" in *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*.

4.2.1 Configuring LDAP, LDAPs, and NPA support

Important

OpenText Information Archive has deprecated support for LDAP and Active Directory (AD). Integration of these communication protocol system will be removed in a future release. Use of LDAP and AD can be done using OTDS. It is recommend to use OTDS instead as soon as possible.

LDAP

Lightweight Directory Access Protocol (LDAP) is an application protocol that aids in searching distributed directory services over IP networks. LDAP is built on a client-server model. LDAP features include access control, support for Unicode, and other functionality that secures resources over IP networks.

IA Web App supports the ability to lock a user whose login attempts have failed multiple times. This ability has been put in place to prevent brute-force attacks.

When the external LDAP locks a user's account, a message asking the user to contact their Administrator to get it unlocked is displayed. Enforcement of this security feature depends on the configuration of the external LDAP or Active Directory.

The following is the account lockout message: `Authentication failed`.

LDAPs

LDAP supports secure (TLS/SSL) access using the LDAPs protocol.

You can use a Non Personal Account (NPA) to access Active Directory and acquire a list of groups, etc.

AD
Active Directory (AD) is a directory service implemented by Microsoft, and is based on LDAP. Active Directory includes hierarchical information about objects on the network where each object can be a computer, printer, user, site, or other related item. Objects have names and sets of attributes. For example, a user inside of an organization can be represented as an object with attributes: First Name, Last Name, Title, Email, Country, etc.

NPA
Non Personal Account, which refers to an anonymous account that can be used to access Active Directory.

You can use these protocols to access Active Directory on a server.

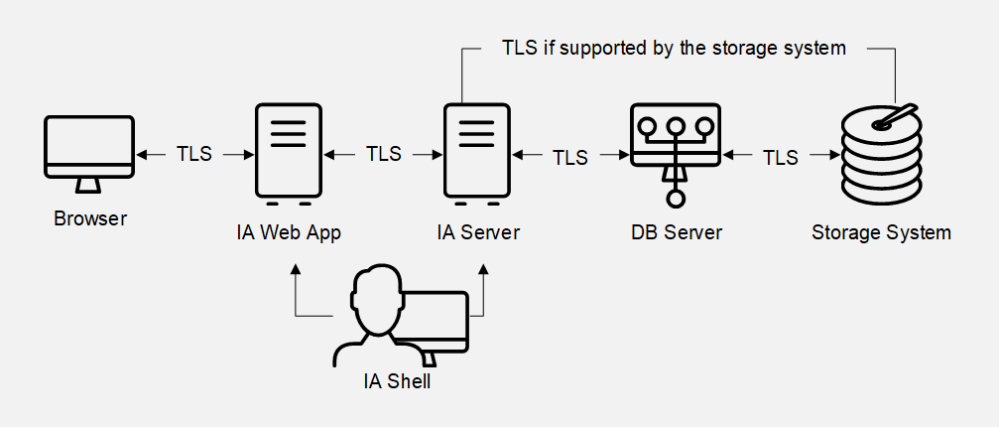
To access LDAP repositories, you must use the `ldap://` URL prefix, which provides non-secure access. To access LDAPs repositories, you must use the `ldaps://` URL prefix, which provides secure access.

As part of this configuration, you must enable at least one example user account with the Administrator role. You log into IA Web App with this example user account and enable the Administrator role for at least one LDAP group. Then you can disable the sample user account.

4.3 Protecting data in transit (SSL)

You can protect the data that is sent between system components and to users' browsers by enabling Transport Layer Security (TLS) connections (TLS is the successor of Secure Sockets Layer, or SSL). OpenText Information Archive uses TLS/SSL to encrypt data and send the encrypted data over mutually authenticated connections to provide privacy and data integrity:

- TLS/SSL connections are private because they use asymmetric and symmetric cryptography for encrypting data. The data cannot be easily decrypted at any point in transit.
- The connection is established with the TLS/SSL handshake, which is secure and reliable, resistant to eavesdropping and man-in-the-middle attacks.
- Communicating parties are identified using public/private key cryptography.
- Data integrity is ensured by message integrity checks. This prevents the alteration or loss of data during transmission.



TLS can be configured for each component independently. If you have multiple DB servers, they can also be configured independently.

With a TLS/SSL connection, OpenText Information Archive components communicate over secure channels. IA Server can connect with the storage system using TLS/SSL, if the storage system supports it. For example, if you are using OpenText Archive Center as storage, it is possible to configure it to use SSL. For more information, see the section 2.2 “Setting up storage for unstructured data” in *OpenText Information Archive - Administration Guide (EARCORE-AGD)*.

4.3.1 Limiting TLS protocol and cipher suites supported by the system

When you configure TLS/SSL, one of the aspects of the initial handshake between the client and server is the selection of supported TLS protocols and cipher suites. The set of ciphers, which are used later to perform cryptography, depends on how the system is configured: whether you use embedded Apache Tomcat (which is part of Spring) or whether you use some external server. Nevertheless, each system supports a number of different protocols and cipher suites out of the box.

When the client sends an initial “hello” to the server, under TLS/SSL and HTTPS, it includes, among other things, a list of the TLS protocols and cipher suites that it supports. The server selects one of the ciphers provided by the client for its response and that suite is used during subsequent communication.

There are several versions of TLS/SSL, starting with SSL 1.0, 2.0 and 3.0, which are outdated and no longer considered secure. Those versions were subsequently replaced with TLS 1.0, 1.1, 1.2, and recently, 1.3. OpenText Information Archive currently supports up to TLS 1.3.

The actual selection of the supported protocols should be decided by each deployment scenario, and largely will consist of a compromise between providing maximum security and at the same time providing adequate support by various potential clients that you want to support. It is recommended that you only support TLS 1.3 to provide maximum security.

The topics below describe how to limit TLS protocols and cipher suites when deploying OpenText Information Archive using embedded Tomcat (which is the out-of-the-box installation). When using external servers, please refer to their documentation about how to limit TLS protocols and cipher suites support.

IA Server and IA Web App must be configured separately to limit TLS protocols and cipher suites supported by the underlying server.

The following list includes some examples of ciphers that system components can use during TLS communication. This is not an exhaustive list. The ciphers available depend on the implementation provided by the OpenJDK release that OpenText Information Archive uses. For more information about the ciphers available, see the documentation for the OpenJDK release.

- Preferred TLSv1.2 256 bits ECDHE-RSA-AES256-GCM-SHA384 Curve P-256 DHE 256
- Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-SHA384 Curve P-256 DHE 256
- Preferred TLSv1.1 256 bits ECDHE-RSA-AES256-SHA Curve P-256 DHE 256
- Accepted TLSv1.1 256 bits DHE-RSA-AES256-SHA DHE 1024 bits
- Preferred TLSv1.0 256 bits ECDHE-RSA-AES256-SHA Curve P-256 DHE 256
- Accepted TLSv1.0 256 bits DHE-RSA-AES256-SHA DHE 1024 bits

4.3.2 Working with digital certificates

A digital certificate is a form of identification used with encrypted connections. Certificates make use of public-key cryptography: a public key encrypts data, which can only be decrypted by using the corresponding private key. The CA (for example, VeriSign) confirms that the owner of the certificate is the owner of the public key.

A self-signed certificate is a certificate that the owner has issued without any validation by a CA. This is not a secure method of identification, and is not recommended for use in production environments, but it can be useful when setting up a demo configuration to test TLS/SSL.

For production environments, it is recommended that each OpenText Information Archive component should have its own certificate signed by a CA. If you have more than one of the same type of component (for example, multiple IA Servers), it is recommended that each distinct component has its own certificate.

4.3.3 Working with keystores and truststores

The keystore for an OpenText Information Archive component stores the component's digital certificate and private key. A truststore stores certificates from trusted clients, which come from CAs. The OpenText Information Archive component uses trusted certificates to establish trust between the component and a client. To trust a certificate, you import the certificate into the truststore.

System components can share a truststore and keystore or use separate truststores and keystores. If you have system components on separate computers, they should not be sharing keystores and truststores unless the components are of the same type (for example, two IA Server instances sharing a keystore and truststore).

4.3.4 Unsupported configurations

All system components must be run in either HTTPS mode or HTTP mode. You can configure a load balancer to run in HTTPS mode, with or without also configuring components to use HTTPS. For more information about setting up load balancing, see Setting up load balancing.

If you deploy IA Web App in HTTPS mode as a WAR web application in Apache Tomcat, then Tomcat must not be configured with the Apache Portable Runtime (APR) connector engine. If you do use HTTPS mode and APR, then the following error message appears in the logs: `Cannot convert access token to JSON`. This message indicates that the client cannot obtain an authentication token to interact with the server.

Chapter 5

Planning a production configuration

By planning a production configuration of OpenText Information Archive, you can help the installation process go more quickly and smoothly. You can make decisions about which computers will host system components, how to manage application traffic, how to configure security, and so on. The following questionnaire is meant to help you make these planning decisions, and the rest of this chapter provides further detail about some of these questions. For more information, see section 3 “What type of archiving should I use?” in *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*.

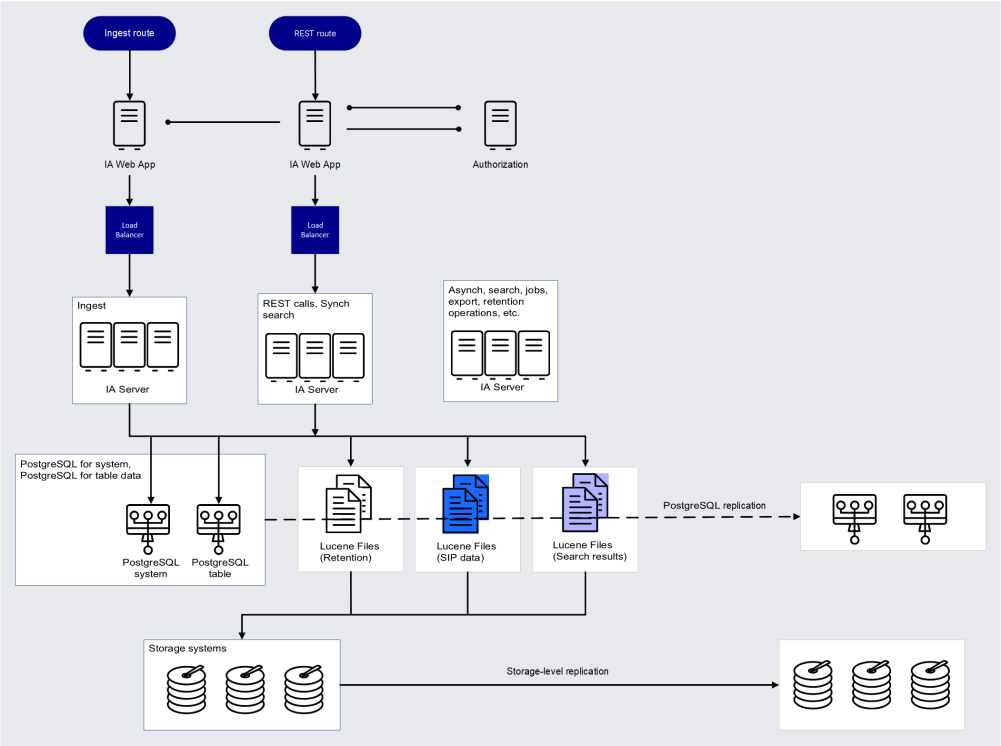
A lot of this detail is about database configuration and disaster recovery setup because they require more explanation and planning and can be difficult to change later on.

Caution

Proper setup is best done before going live as it can be difficult to change later on.

5.1 Typical architecture

The following diagram gives an example of what a production system might look like



Customers will likely use multiple instances of IA Web App and the IA Server and use a load balancer to distribute the work. Note that it is possible to specialize IAS servers (as shown in the diagram to be focused for ingestion or searches). This specialization is done through configuration to not perform certain actions.

The structured data for table application is stored in PostgreSQL (PG) as well as the system data.

Lucene is used for retention data, SIP data, and search results and must be stored on a file system.

The file system used for Lucene must be read-write accessible to all IA Server nodes and therefore typically needs to be either a network-mapped mount point or shared from whichever node it is hosted. In case one or more IA Server nodes cannot access this file system, there will very likely be unpredictable accessibility problems whenever interacting with the file system. The file system is referred to by the configured file system root(s).

Finally, there are considerations for what sort of replication you want to configure for PostgreSQL and for the storage level.

5.2 Installation questionnaire

Before you install OpenText Information Archive, consider the following questions:

- How many computers do you want to use? Do you want to install multiple instances of components? Do you want to set up load balancing?
- What is your ETL process? What system do you want your archived data to come in from?
- Do you want to encrypt the passwords that are stored in configuration files?
- Do you want to protect data in transit with encryption? Do you want to encrypt your archived data?
- Do you want to ingest data from table archiving and SIP archiving applications at the same time?
- Do you want to install system components as system services? Or would you rather start and stop the services manually using a command prompt or your startup scripts called by your enterprise monitoring and scheduling tool?
- Do you want to use one user account in Linux to install OpenText Information Archive, and another to run the system services?
- How do you want to authenticate users? Do you want to use OpenText Directory Services (OTDS) in single sign-on (SSO) mode? This requires you to encrypt data in transit.

  For more information, see section 7.1.1 "Authenticating users – configuring OpenText Directory Services" in *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*.
- Do you want to replicate your data to a disaster recovery site?
- How do you plan to use PostgreSQL? Do you want a DBA to pre-create databases?

5.3 Planning a PostgreSQL server configuration

Planning a PostgreSQL server configuration is possibly the most important part of planning a production deployment because it involves understanding some additional underlying concepts and requires the most planning ahead.

You should ideally develop expertise in configuring, optimizing, and administering PostgreSQL. For more information, there are guides available online since it is an open source product.

5.3.1 PostgreSQL data nodes, databases and setup

For PostgreSQL, you need a server to host databases. OpenText Information Archive refers to this as a data node (PostgreSQL refers to this as a data cluster, both terms may be used interchangeably), which is a logical container for the database hosted by that server.

The system requires a setup of at least one data node for a demo configuration, and at least two data nodes for a production configuration (one for system data and at least one for structured data of table applications).

You can have each data node created by the OpenText Information Archive setup script (although this is typically only used for Demo deployments), or create them yourself and let setup/configuration know about them.

Caution

OpenText is not responsible for further maintenance/support on PostgreSQL data nodes, including those created by OpenText Information Archive setup itself.

Note: If you instruct OpenText Information Archive to create the database cluster, you will need to review the generated PostgreSQL configuration files, which will be created in the `<IA_ROOT>/psql/conf` directory.

Important

OpenText Information Archive setup is only capable of reliably creating PostgreSQL data nodes on local file systems. When creating a PostgreSQL data node through the setup, it is necessary to specify a data directory that is on the local machine's file system. Specifying a data directory on a mapped network drive is likely to fail during the initialization/copy step. For performance reasons, it is also better to use a local file system for the PostgreSQL data node data directory.

If, however, a remote network drive needs to be used, it is recommended to create such a data node manually without using setup.

It is also possible to create the empty data node on a local file system using setup and move it to a remote network drive afterwards, but this also requires manual reconfiguration of the PostgreSQL data node, so taking full ownership of the data node might be a better option.

For IA Server to work with these data nodes, it needs to have at least the following configured per data node:

- A user (for now, referred to as a data node user) with a name and password.
- A connection URL.

The data node user is used by IA Server when:

- Creating or dropping databases,
- Checking a database existence, or
- Providing information on the PostgreSQL version, *etc.*.

Note: The planning of data nodes is not only about which physical host runs which database cluster instance. It is also about where the data files are stored.

OpenText Information Archive requires several databases to be created in its data nodes (see below to learn which databases).

For databases within a data node, you have a similar choice to either let OpenText Information Archive create them or pre-create them in your data node(s) yourself. If you let OpenText Information Archive create the databases, IA Server will do this using a connection to the data node with the configured data node user. Therefore, if you want this option, the data node user you provided requires both `CREATEDB` and `CREATEROLE` permissions. If you create the data nodes yourself, the configured data node user only needs the LOGIN privilege, except for the following:

Note: The feature to cache out and cache in table applications can only be supported when the configured PostgreSQL data node user has `CREATEDB` permission to both delete and create databases. Because OpenText Information Archive is unable to determine the permissions of configured PostgreSQL users, it is not possible to disable this feature preemptively and, if attempted, the process will fail.

For every database, IA Server needs to have a user with a name and a password, which we will refer to as the database user (as opposed to the data node user).

The database user is used by IA Server when:

- Creating tables
- Running search/update queries, *etc*.

If you choose to create the databases yourself, you should also create the database users. The database users need to be configured to be the database owners of their database. Other than that, they need to have `LOGIN` privileges. For more information, see Pre-creating databases in a PostgreSQL node.

Databases, database owners and their passwords, similar to data nodes, data node users and their passwords, need to be registered with OpenText Information Archive. For example, IA Server can read their configurations on startup from a YAML file, which in turn can be created by the setup script to reflect your answers to its questions.

Important

When configuring PostgreSQL users/roles to be used by OpenText Information Archive to connect to data nodes and databases, it is recommended to use all lowercase characters for them and no hyphens (`-`) or other special characters other than underscores. If you let OpenText Information Archive create users/

roles with uppercase characters dynamically for you in PostgreSQL, subsequent connection attempts will fail, as PostgreSQL stores their names as lowercase. If you configure users/roles with hyphens, then, for example, configuring that role to be the owner of a database, will fail with syntax error.

5.3.1.1 Pre-creating databases in a PostgreSQL node

To pre-create databases in a PostgreSQL node:

1. Create the following databases in the System PostgreSQL data node and create the database owner account for these databases:

   - `system`
   - `rollforward`
   - `audit`
   - `synchronization`

   To ensure correct search and sorting behavior, it is strongly recommended to make all PostgreSQL databases use UTF-8 encoding. For the system data database, this is important enough to have new installations no longer accept a system database with another encoding. For existing installations, a warning will be logged during every startup when a different encoding is detected for at least the system data database. Using a different encoding for any PostgreSQL database may result in unexpected or incorrect search behavior and sorting may result in unexpected ordering. It is not possible to change database encoding once a PostgreSQL database is created. Instead, a new database needs to be created and all data would need to be migrated over while all IA Servers are shutdown.

2. Create the following databases in the structured data PostgreSQL node:

   - `Reports-rdb`

   A database for table applications must also be created. It is important to note that these must be created *before* installing the applications using declarative configuration.

Note: The feature to cache out and cache in table applications can only be supported when the configured PostgreSQL data node user has `createDB` permission to both delete and create databases. Because OpenText Information Archive is unable to determine the permissions of configured PostgreSQL users, it is not possible to disable this feature preemptively and, if attempted, the process will fail.

Specifically related to application structured data databases, the feature to cache out and cache in table applications requires OpenText Information Archive to both delete and create the database(s) associated with the application. When these databases need to be pre-created due to lack of required permissions; otherwise, this feature will not work and the applications cannot be cached out and/or in.

Because it is not possible to detect available permissions, this feature cannot be preemptively disabled but will, instead, fail when attempted.

5.3.2 Running the system with a different version of PostgreSQL than the one included in the distribution

IA Server uses PostgreSQL client (command line) software for backing up and restoring table application databases. Therefore, it must be able to find this software on the machine on which it runs. It is vital that the version of the PostgreSQL client software is the same as the PostgreSQL servers' version.

If you use setup to create/configure everything (again, this is typically only used for Demo deployments), you do not need to worry about this. However, if you pre-created your data nodes, and they have a different version from the PostgreSQL shipped with your OpenText Information Archive distribution, install a version of the PostgreSQL on the IA Server machine that matches the versions of your PostgreSQL data nodes. In addition, you need to ensure the following environment variable points to your local PostgreSQL (client) installation:

```
IAPSQL_HOME
```

If you change this environment variable after IA Server is started, you must restart IA Server for the change to take effect. If you encounter any issues, see Cannot take a backup of a table database or take a table application offline.

5.3.3 System requirements for data nodes

OpenText Information Archive stores several different types of data in PostgreSQL:

- System data
- Structured (ingested) data for table archiving OpenText Information Archive applications

By default, a demo configuration stores all the data above in one data node.

In a production environment, because of more demanding requirements for availability, scalability, performance, flexibility, and disaster recovery, consider setting up at least two data nodes, and possibly more:

1. *System data*
2. *Table application structured (ingested) data*:
    - You might want to create separate data nodes for different table applications or groups of applications. This can make it easier to manage data distribution for large amounts of data. You should separate any data nodes for business data if they conflict with each other with long transactions (for example, the indexing of large tables) and high-volume, frequent transactions.
    - Otherwise, you can set up a single data node for all structured data.

Note: These are planning decisions that you should make before you install OpenText Information Archive because migrating to a deployment with additional data nodes is not straightforward.

5.3.4 Determining disk capacity for storing PostgreSQL data files

It is important to allocate sufficient disk space for the various types of data that OpenText Information Archive stores in the PostgreSQL server, and to monitor disk usage to prevent the database from outgrowing its allocated disk volumes.

The system database is typically the smallest of the two databases mentioned earlier (the other one being the structured data), at under 100 GB, but can possibly grow quite large. The primary reason for such growth is having a lot of table or SIP applications (100 or more). Consequently, the system-related information grows.

Also, some parts of the system data expand in bursts (for example, when running large background operations). You will see the PostgreSQL data files grow quickly, and then not shrink anymore. However, the IA Server does ensure that any unused space within these files can be reused by the PostgreSQL server.

5.3.5 PostgreSQL connections and connection pooling

By default, the PostgreSQL Server uses a `max_connections` of 200, which may need to be increased based on the expected load of the environment and total number of databases that will be accessed concurrently.

5.3.5.1 System data

For the PostgreSQL node for system data, each IA Server maintains four connection pools (OpenText Information Archive system data is divided over four databases in the PostgreSQL system data node). Each of those pools has a maximum size of 100 connections (this number is configurable in the IA Server `application.yml` file), and an initial size of 4. This means that the maximum of active connections to the system data node is 4 x 100 x [number of IA Server nodes].

The used connection pool logic has multiple means to control the number of active connections to the PostgreSQL Server.

First, it is possible to configure the maximum active connections in the pool. Each connection pool will never have more connections than the value set. If the value is not high enough, logic will block and potentially time-out.

Unused idle connections that have been released back to the pool will remain open for some time before being closed completely as part of core connection pool behavior.

It is, however, possible to limit the number of idle connections by means of `maxIdle`. Active connections that are released when the number of idle connections in the connection pool exceed this number will be closed faster than normal.

> **Important**
> It is also possible to enable more thorough cleanup of abandoned connections by means of the `removeAbandoned` parameter. This, however, should remain **disabled** at all times, unless there is a very explicit need to enable it, as it will break the IA Server logic and introduce problems that prevent IA Server from functioning properly.
>
> Under normal circumstances, there should be no need to remove abandoned connections, as these typically only exist due to either bugs in IA Server code, or highly unusual situations that are difficult or near impossible to anticipate properly.

If you decide to install PgBouncer in front of the PostgreSQL server, be aware that, currently, only session mode (see PgBouncer documentation for what this implies) can be used.

Configurable pool options in the `application.yml` file:

- `systemData.psql.pool.maxActive`
- `systemData.psql.pool.maxIdle`
- `systemData.psql.pool.initialSize`
- `systemData.psql.pool.removeAbandoned`
- `systemData.psql.pool.removeAbandonedTimeout`

5.3.5.2 Structured data

For structured data, IA Server has a pool for each database (we have one PostgreSQL database per table application). The following are the default values for pool management:

- `maxActive: 25`
- `maxIdle: 4`
- `initialSize: 1`
- `removeAbandoned: false`
- `removeAbandonedTimeout: 300 (seconds)`

How this affects PostgreSQL servers you deployed for structured data, depends on how many table applications you have, how many structured databases you put into one server, and how intensively these are used (especially for search and certain retention operations).

If you decide to install PgBouncer in front of the PostgreSQL server(s) for structured data, be aware that, currently, only session mode (see pgbouncer documentation for what this implies) can be used.

5.4 Environment setup

For any Linux machines that will host an IA Server, you will need to increase the maximum number of open files. The default value is `1024` and it is recommended to set it to `100000`. The `ulimit` command can be used to see the current value and update the value using the appropriate command for your Linux version. This change goes into effect the next time the IA Server starts.

Another system property that you may need to change is `vm.max_map_count`. The memory mapped files are used extensively by Lucene indexes and the default value of 64K may not be enough for your setup. When Lucene is unable to memory map a file, it throws an IOException, which is logged to the `errors.log` (`java.io.IOException: Map failed: MMapIndexInput(path="path-to-a-lucene-index-file"`). The property can be set directly in `/etc/sysctl.conf` or by using the `syctl` command. For example:

```
linuxConfig:
   sysctl:
     vm.max_map_count: 262144
```

5.4.1 Additional performance considerations

For best performance, leave the working directories for each system component on its local disk (for example, the ingest directory, the receiver directory, and the Tomcat temp directory). Storing these directories on network storage might cause performance and scalability issues. You should keep this in mind when you install the IA Web App and when you create and configure applications.

5.5 Protecting sensitive data

The following are required to configure the system:

- Need to protect sensitive information in configuration files.
- Need to configure how to protect the sensitive information in the system database.
- Need to protect the application data.

If you choose to protect sensitive information in configuration files, you have the following options:

- Vault
- AWS Secrets Manager
- Gemalto
- Use the product's encryption functionality (for more information, see section 1 "Overview" in *OpenText Information Archive - Encryption Guide (EARCORE-AGE)*)

5.5.1 Integration with Vault

OpenText Information Archive supports integration with HashiCorp Vault. Each system component: IA Server, IA Web App, IA Shell can be integrated with Vault. That means that all secrets/passwords required for a specific component's initialization, currently found in configuration files (either YML or properties), can be moved to Vault. This is an opt-in process which, by default, is not enabled. When enabled, system components reach out to a configured Vault instance to gather all secrets and passwords, and merge those with the rest of the configuration properties.

The integration between OpenText Information Archive and Vault can be broken into four separate phases:

- Generation of Vault JSON templates containing secrets/passwords per component
- Configuration of Vault instance
- Populating Vault with secrets/passwords

5.5.1.1 Authentication with Vault

OpenText Information Archive supports multiple ways of authenticating with Vault:

`NONE`
: In this mode, you disable authentication for system components and, instead, point them at the Agent Proxy. Agent Proxy is a Vault Agent, running as a proxy between Vault Server and the system components. Agent Proxy needs to be configured with some type of authentication access to the Vault Server (for example, AppRole) and set for no-authentication access. For more information on the Vault Agent API proxy see the Hashicorp developer website. This is the out-of-the-box default.

`TOKEN`
: This is the simplest way of authentication with Vault. All you need to provide is a valid token generated by the role with permissions for the locations where all secrets are stored. Once provided to OpenText Information Archive configuration, each component will be able to authenticate/login (to Vault) and fetch all necessary secrets and passwords. However, the tokens have a expiry date and, additionally, this method of authentication does expose Vault's token in plain text view in the configuration file, so token itself can be compromised.

`APPROLE`
: This requires provisioning a special role in Vault with access to all required secrets, and then providing a role ID and secret ID to OpenText Information Archive configuration. Each component, at run time, uses these credentials to login to Vault and obtain its own token. This way, you do not have to worry about a token's expiry. However, since a role's (both) credentials are stored in the configuration files, they can be compromised.

5.5.1.2 Configuration of Vault

Full configuration of Vault is beyond the scope of this document. This section focuses on what configuration is expected from a given Vault instance to effectively enable integration with OpenText Information Archive.

- Secrets Engine – OpenText Information Archive supports KV Secrets Engine (both: V1 and V2). The KV engine needs to be enabled on a Vault instance. When enabling the KV Secrets Engine, make a note of the path, which is, by default `kv`, but could be anything like `secret`, *etc*. This value will be needed later when configuring the Vault profile and corresponds to the `vault.kv.backend` key. For some of the examples that follow, assume that the path has been set as `secret`.
- At this point you have a running Vault instance with enabled KV Secrets Engine. You will populate Vault with actual values later, when you have generated the JSON templates.
- Ensure appropriate authentication is enabled.
- The IA Shell component is referred to interchangeably as `shell` or `cli`, they both mean the same component representing command line interface or shell integration of OpenText Information Archive.

5.5.1.3 Additional step when using on premises deployment with Vault and OTDS

The setup utility that creates JSON templates for Vault is not OTDS aware:

- For the IA Web App, the JSON template does not include the following secrets in the `infoarchive.gateway.profile.OTDS` profile:
  - `OTDS.infoarchive.clients.gateway.clientSecret`
  - `OTDS.infoarchive.clients.iawa.clientSecret`
  - `OTDS.password`
- For the IA Server, `OTDS.password` is not present in the JSON template. This is a password for OTDS that is part of the `otds` profile for the IA Server.

Complete the following two procedures.

To manually update the configuration of the IA Web App:

1. In the `<IA_ROOT>\infoarchive\config\iawebapp\application-infoarchive.gateway.profile.OTDS.yml` profile, copy the values for the `OTDS.password`, `OTDS.infoarchive.clients.iawa.clientSecret`, and `OTDS.infoarchive.clients.gateway.clientSecret` keys.
2. Insert the three key/value pairs in the `<IA_ROOT>\infoarchive\config\vault\iawebapp.json` template, using commas to separate the entries. For example, if your original JSON template looked like the following:

```
{
        "infoarchive.gateway.token.secret": "XXXX"
}
```

It may look similar to the following after inserting new key/value pairs:

```
{
        "OTDS.infoarchive.clients.gateway.clientSecret": "XXXXXXXXX",
        "OTDS.infoarchive.clients.iawa.clientSecret": "XXXXXXX",
        "OTDS.password": "XXXXXXXXXXXXX",
        "infoarchive.gateway.token.secret": "XXXXXXX*"
}
```

3. Save the `iawebapp.json` template. Reset the three keys in the `application-infoarchive.gateway.profile.OTDS.yml` template (set these three keys to empty values), then save the file.

To manually update the configuration of the IA Server:

1. In the `<IA_ROOT>\infoarchive\config\iaserver\application-otds.yml` OTDS profile, copy the value of the `OTDS.password` key.
2. Insert the value of the `OTDS.password` key in the `<IA_ROOT>\infoarchive\config\vault\iaserver.json` template, using a comma to separate the entries. The original JSON template looked like the following:

```
{
  "infoarchive.gateway.token.secret": "XXXXXXX*",
  "crypto.keyStore.keyStorePass": "XXXXXXXX",
  "synchronizationData.psql.database.admin.password": "XXXXXXXX",
  "systemData.psql.databaseCluster.superuser.password": "XXXX",
  "auditData.psql.database.admin.password": "XXXXXXXX",
  "rollForwardData.psql.database.admin.password": "(XXXXXXXX",
  "systemData.psql.database.admin.password": "XXXX"
}
```

After inserting the `OTDS.password` key, the JSON template would look like the following:

```
{
  "OTDS.password": "XXXXXXXXXXXX",
  "infoarchive.gateway.token.secret": "XXXXXXX*",
  "crypto.keyStore.keyStorePass": "XXXXXXXX",
  "synchronizationData.psql.database.admin.password": "XXXXXXXX",
  "systemData.psql.databaseCluster.superuser.password": "XXXX",
  "auditData.psql.database.admin.password": "XXXXXXXX",
  "rollForwardData.psql.database.admin.password": "(XXXXXXXX",
  "systemData.psql.database.admin.password": "XXXX"
}
```

3. Save the `iaserver.json` template. Remove the `OTDS.password` value in the `application-otds.yml` file and save your change.

5.5.1.4 Generating Vault JSON templates

When you run the OpenText Information Archive setup and answer questions related to Vault, the setup utility automatically creates the necessary Vault JSON templates.

5.5.1.5 Populating Vault with secrets

Once you have your Vault JSON templates, back to the Vault's interface and go to the KV Secrets Engine, as it can be now populated with JSON data.

- When using Vault's interface, go to Secrets and, from that view, select KV mounting path (in this scenario it is `secret`. That takes you to the Secret Configuration screen. Create a new secret, since you are preparing a location for secrets for a specific customer, choose a name for the path that is meaningful (this scenario uses `iacustomer`). This location should correspond to precisely a location chosen in the previous step when you created a role and policy for the `infoarchive` client.

  – The secrets can be created in JSON or simple properties format. For now, select JSON.

  – The customer's name becomes the first part of the actual path to the secret's location.

  – Under the same path, create three actual folders, each corresponding to one of the system components.

    The first path is `iacustomer/ias`, which this denotes the IA Server folder within the `iacustomer` path. For the actual data, remove anything that was pre-populated in the Data section of the interface screen by Vault (for example, remove curly brackets, *etc*.), and then copy contents of the `iaserver.json` template and paste it to the Data portion of the interface.

    Vault validates JSON for errors so, if the copy/paste action resulted in some issues, such as incorrect formatting, it has to be corrected before you can successfully save it. Once all potential JSON issues are resolved, save your secret.

  – Go back to the Secrets view and select the secret KV path (or alternatively click secret part of the breadcrumb view shown in upper-left portion of the interface). Create another secret and, this time, choose path `iacustomer/shell`, populate it with the content of `iashell.json` then save it.

  – Go back to the secret again and create two more secrets: `iacustomer/iawa` and, this time, use `iawebapp.json`, then save it.

    Keep track of these paths as they will be needed later when configuring vault profiles for each of the components: IA Server, IA Web App and IA Shell (for more information, see Configuration of Vault). They will correspond to the properties under the `vault.kv.application` key: `ias`, `iawa`, `shell` properties.

5.5.1.6 Vault Agent API proxy

The out-of-the-box recommended configuration for integration with Vault is to use Agent Proxy.

You need to deploy Agent Proxy on each machine where a system component is running, and configure appropriate authentication so that the agent can connect to the Vault with the appropriate credentials. Typically, those will involve setting it up to use AppRole integration and re-using the `infoarchive` role that was created earlier. This ensures that the agent has access to all of OpenText Information Archive's relevant secrets inside the Vault. At the same time, enable no-authentication access to the Agent itself. This allows OpenText Information Archive components to connect to the Agent without any type of token. Agent will act as a proxy between system components and the Vault Server.

Refer to the HashiCorp site (https://developer.hashicorp.com/vault/docs/agent/apiproxy) for more information.

5.5.1.7 Vault configuration

The Vault profiles for each component can be found in `<IA_ROOT>/infoarchive/config` locations for each respective component: `iaserver`, `iashell` and `iawebapp` folders.

Each system component connecting to Vault has a special vault profile containing all the Vault related details. These are:

```
spring.cloud.vault.authentication=NONE
```

Sets the authentication to NONE as you are connecting to Vault Agent directly, which has been configured for `no-auth` access.

```
spring.cloud.vault.fail-fast=true
```

Set to true – in case there are any issues interacting with the Vault, you receive feedback immediately.

```
spring.cloud.vault.kv.enabled=true
```

OpenText Information Archive supports the KV engine so you need to ensure it is also enabled in the configuration.

```
spring.cloud.vault.kv.backend=secret
```

This is the value of the path component when enabling KV engine in the Vault. Copy that value from Vault's configuration screen for KV engine.

```
spring.cloud.vault.kv.application-name=iacustomer/iaserver
spring.cloud.vault.kv.default-context=iacustomer/iaserver
```

Both `application-name` and `default-context` should point to the location inside the KV engine where you stored the secrets. Typically, at the root of the KV engine, you create one folder for the customer, in this case it is `iacustomer`, and there you create three subfolders, one for each of the specific system components: IA Server, IA Web

App, and IA Shell. For example, for the IA Server component, the value would be `iacustomer/iaserver`. For IA Shell, it would be `iacustomer/shell`. For the IA Web App, `iacustomer/iawa`, etc. Ensure these names match the locations and paths in the KV engine in your Vault.

```
spring.cloud.vault.uri=http://localhost:8100
```

This needs to point to the location of Vault Agent, which would typically run on `localhost/127.0.0.1`, as it should be local to each component. All components, if running on the same machine, can share Agent Proxy. While Agent Proxy typically runs on `127.0.0.1` on `port 8100`, you should adjust the port if you configured it to run on a different port.

```
spring.config.import=vault://
```

Leave this value as is.

5.5.1.8 Starting each component

When using Vault integration, all secrets/passwords from typical YAML configuration files are moved to Vault and YAML configuration will not contain values for any secret keys. Those values will be fetched from Vault at runtime, during component's startup. Each component connects to a locally running Agent Proxy, which forwards each request (for passwords/secrets) to the Vault. For example, IA Server needs to fetch credential details for database access from the Vault. If successful, it will fetch the credentials, then is able to connect and configure PostgreSQL databases, as required.

If a component, such as IA Server, fails to fetch the credentials, ensure:

1. The component can connect to the Agent Proxy. If the component cannot connect to the local Agent Proxy, there should be an error message similar to

   ```
   "Connection refused: connect" (refer to the example below: Caused by:
   org.apache.http.conn.HttpHostConnectException: Connect to localhost:8100 [localhost/
   127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Connection refused: connect
   ```

   Check the Vault profile for the component (for example, `<IA_ROOT>/infoarchive/config/iaserver/application-vault.profile`) for the value of `spring.cloud.vault.uri` key. It should be set to the URL of the local Agent Proxy (for example, `spring.cloud.vault.uri=http://localhost:8100`. Ensure that host/port are configured correctly to point to your local Agent Proxy's address. Make any necessary adjustments and restart the component.

2. Check to ensure Agent Proxy is receiving requests from your component and can forward them to the Vault. Check Agent Proxy (output or log file, *etc*) for an indication that Agent Proxy was forwarding requests for the component. You should see a message similar to:

   ```
   received request: method=GET path=/v1/kv/data/iacustomer/iaserver
   ```

   Or:

   ```
   forwarding request to Vault: method=GET path=/v1/kv/data/iacustomer/iaserver
   ```

Check to ensure that paths correspond correctly to your configuration in Vault's profile. The path typically has the following structure: `/v1/<backend>/data/<application-name>`. You may see more than one request being forwarded due to the fact that Spring tries to append all active profiles to the path, which is expected. At least one of the forwarded requests should point exactly to the location as configured in Vault's profile. You will also need to ensure that Agent Proxy is correctly configured to point to the Vault server and that is uses appropriate authentication that not only allows it to login to the Vault, but that it also can successfully access your component's secrets using said authentication.

3. If the component can connect to the Agent Proxy, and Agent Proxy is forwarding requests to the Vault, if Agent Proxy does not have correct permissions to access your secrets, you may still get errors in the component's log, similar to the following:

```
Caused by: org.springframework.web.client.HttpClientErrorException$Forbidden: 403
Forbidden: "

{"errors":["1 error occurred:\n\t* permission denied\n\n"]}
<EOL>"
```

This indicates that Agent Proxy does not have enough permission to access the secrets. Double check Vault's profile (keys such as `spring.cloud.vault.kv.backend`, `spring.cloud.vault.kv.application-name`, or `spring.cloud.vault.kv.default-context`) to ensure they are correctly configured. Also, double-check the combined path found in Agent Proxy's output to ensure it points to your component's secrets. Lastly, ensure the authentication method Agent Proxy is using to login to the Vault has proper access to the component's location where secrets are stored.

When all these issues are corrected, you should see your component starting up properly with no errors.

5.5.2 Integration with Google Cloud Platform® service GCP

Like Vault, integration with GCP Secret Manager is optional. The opt-in model is based on the Spring profile.

The `gcp-secrets-manager` profile is similar to a Vault profile, and contains a small set of properties that allow each component to connect to Secret Manager and read configured secrets.

Example 5-1:

```
spring.cloud.gcp.secretmanager.enabled=true
spring.cloud.gcp.core.enabled=true
spring.cloud.gcp.secretmanager.allow-default-secret=true
spring.cloud.gcp.project-id=<GCP project ID>
spring.cloud.gcp.secretmanager.project-id=<GCP project ID>
spring.config.import=sm@
systemData.psql.database.admin.password=${sm@systemData_psql_database_admin_password}
crypto.keyStore.keyStorePass=${sm@crypto_keyStore_keyStorePass}
…
infoarchive.gateway.token.secret=${sm@infoarchive_gateway_token_secret}
OTDS.password=${sm@OTDS_password}
```

```
OTDS.infoarchive.clients.server.clientSecret=$
{sm@OTDS_infoarchive_clients_server_clientSecret}
```

In the example above, the first six sets of properties are configuration properties that control whether GCP Secret Manager integration is enabled, whether to allow default secrets, and provide the ID of the GCP service.

The last configuration property (`spring.config.import`) contains a default prefix for the secrets stored in GCP Secret Manager. It is recommended that you keep it as it is displayed in the example above.

All these configuration properties will be the same for each OpenText Information Archive component: IA Server, Gateway, or IA Shell.

What follows these initial set of configuration-level properties are the references to actual properties dedicated to each component. Each component has a separate set of properties.

5.5.2.1 Configuration in Helm's values file

The following section of Helm's values file is dedicated to configuring the GCP Secret Manager:

Example 5-2:

```
gcpSecretsManager:
  # set to true if using integration with GCP SM
  enabled: true
  #
  allowDefaultSecrets: true
  # Supported authentication methods: envVar
  # For envVar type: GOOGLE_APPLICATION_CREDENTIALS env variable will have json access
key
  authentication: "GOOGLE_APPLICATION_CREDENTIALS"
  # project id - most times gcp and secret manager project ids will be the same
  projectId:
  secretmanager:
    projectId:
  # Google application credentials, this should be set externally as it is a JSON file.
  accessKey:
  # Prefix for all the secrets, default is empty. Set to some value, i.e. customer name,
  # to allow for multiple deployments in the same SM. Use only alphanumeric characters,
  # preferably end with an underscore, i.e. acme_
  secretsPrefix:
```

Currently, the only authentication scheme supported is `GOOGLE_APPLICATION_CREDENTIALS`.

For most deployments, `projectId` and `secretmanager.projectId` will be the same.

Set the `accessKey` property to reference the JSON file that contains the JSON representation of the GCP service account's key that accesses the GCP Secret Manager, which contains secrets for your Information Archive deployment. The

service account requires appropriate access to the GCP Secret Manager. At the very least, you need to assign the Secret Manager Secret Accessor Role to the service account; otherwise the service account will not have access to read secrets from Secret Manager.

By default, the `secretsPrefix` property can be left as is for most deployments. If you use the same Secret Manager to manage secrets for multiple OpenText Information Archive deployments, ensure each deployment has a unique prefix. For example, if Secret Manager is shared for two deployments – those could be two deployments for the same customer or for two different customers – in each case, the prefix should be unique for the deployment. You could use the prefix: `acme_stage_` for one deployment and `acme_prod_` for the other deployment.

5.5.2.2 Storing secrets in GCP Secret Manager

Use the Password Generation utility as you normally would to generate all the secrets. Notice that one of the new artifacts generated by the utility is a text template (`gcp_secret_manager.txt`). This template contains all the secrets that need to be stored in GCP Secret Manager .

The template contains a list of key-value properties in which the key corresponds to the name of the secret and the value contains the value of the secret itself.

Currently, secrets can only be created using the Google API or Google UI. Use either of these methods to create secrets. Both methods allow you to create one secret at a time.

Note: If you are using `secretsPrefix`, add the prefix to each name of the secret. For example, instead of creating a secret with the name `OTDS_password`, if the `secretsPrefix` property is set to `acme_prod_`, the name of the secret would be `acme_prod_OTDS_password`. Repeat this for every secret that needs to be created.

The structure of the template itself can easily be used by a script if you were going to use Google API to automate the process of secret creation.

Since the template contains actual passwords/secrets, ensure you protect access to it and delete the file (or archive, *etc.*) if you are done creating secrets.

5.5.3 Integration with AWS Secrets Manager

Like Vault, integration with AWS Secrets Manager is optional. Opt-in is based on the Spring profile. When running setup, indicate that you want to use Secrets Manager to store secrets that would otherwise be stored in the various configuration files stored on disk. This feature only applies to those secrets. It does not apply to any secrets stored as `SystemData` in PostgreSQL.

During setup, two framework questions are asked to determine the region of AWS to use and the location root that will be shared between all system components using Secrets Manager. Also, for each component, you will be asked to enter a `Secret name`, which identifies the collection of secrets stored in AWS (for that component). For more information on AWS Secrets Manager see the Amazon AWS website.

The `aws-secret-manager` profile is similar to a Vault profile, and contains minimal set of properties that allow each component to connect to Secrets Manager and read configured secrets.

Example 5-3:

```
aws.secretsmanager.enabled=true
aws.secretsmanager.failFast=true
spring.cloud.aws.region.static=us-east-1

spring.config.import=aws-secretsmanager:prod1/iacustomer/server
```

`aws.secretsmanager.enabled / aws.secretsmanager.failFast / spring.cloud.aws.region.static properties`

- The `aws.secretsmanager.enabled` property allows you to allow Secrets Manager integration.
- The `aws.secretsmanager.failFast` property allows you to configure components to quickly fail if there are any issues accessing Secrets Manager.
- The `spring.cloud.aws.region.static` property allows you to configure regional settings.

`spring.config.import property`

This property contains the actual path to the secrets (preceded by the `aws-secretmanager:` prefix). The path itself (`prod1/iacustomer/server` in Example 5–1 above) should correspond exactly to the secrets path (referred sometimes as a secret name), as defined in Secrets Manager.

The secret value is a JSON structure and looks exactly like the secret value for HashiCorp Vault:

Example 5-4:

```
{
  "OTDS.password": "otds",
```

```
  "crypto.keyStore.keyStorePass": "NgVS5IgfbM",
  "infoarchive.gateway.token.secret": "YJe4Yaikngs",
  "systemData.psql.database.admin.password": "Ga#Rd#1RT7t9ogGdI7Rm",
  "systemData.psql.database.connectionProperties.sslpassword": "",
  "systemData.psql.databaseCluster.connectionProperties.sslpassword": "",
  "systemData.psql.databaseCluster.superuser.password": "Ga#Rd#1RT7t9ogGdI7Rm"
}
```

As with Vault, Secrets Manager breaks the secret values down to key-value pairs.

5.5.3.1 Importing secrets

As with Vault, you can determine any preferred method of importing secrets into Secrets Manager, as long as they are ingested using the expected JSON templates. These templates are the same as for Vault integration.

Example 5-5: IA Server secrets as JSON

```
{
  "OTDS.password": "otds",
  "crypto.keyStore.keyStorePass": "XXXXXXXXXX",
  "infoarchive.gateway.token.secret": "XXXXXXXXXXX",
  "systemData.psql.database.admin.password": "XXXXXXXXXXXXXXXXXXX",
  "systemData.psql.database.connectionProperties.sslpassword": "",
  "systemData.psql.databaseCluster.connectionProperties.sslpassword": "",
  "systemData.psql.databaseCluster.superuser.password": "XXXXXXXXXXXXXXXXXXX"
}
```

Example 5-6: IA Web App secrets as JSON

```
{
  "OTDS.infoarchive.clients.gateway.clientSecret": "XXXXXXXXXX",
  "OTDS.infoarchive.clients.iawa.clientSecret": "XXXXXXXXXX",
  "OTDS.password": "otds",
  "infoarchive.gateway.client.ssl.key-store-password": "",
  "infoarchive.gateway.client.ssl.trust-store-password": "XXXXXXX",
  "infoarchive.gateway.token.secret": "XXXXXXXXXXX"
}
```

Example 5-7: IA Shell secrets as JSON

```
{
  "connection.clientSecret": "XXXXXXXXXX",
  "connection.userPassword": "XXXXXXXXXX",
  "rdbDataNode.connectionProperties.sslpassword": "",
  "rdbDataNode.password": "XXXXXXXXXXXXXXXXXXX",
  "rdbDatabase.connectionProperties.sslpassword": "",
  "rdbDatabase.password": "XXXXXXXXXXXXXXXXXXX",
  "ssl.trustStorePassword": "XXXXXXX"
}
```

Example 5-8: OTDS secrets as JSON

```
{
  "OTDS": {
    "basicOauth2ClientsData": {
      "oauth2Clients": [
        {
          "name": "infoarchive.gateway",
          "secret": "Y5FrWOvM6c"
        },
        {
          "name": "infoarchive.iawa",
          "secret": "XsLexOOwIv"
        },
        {
          "name": "infoarchive.cli",
          "secret": "D1Z1pKFgnK"
        },
        {
          "name": "infoarchive.jdbc",
          "secret": "kKPHkhZG1u"
        },
        {
          "name": "infoarchive.sap",
          "secret": "kKPHkhZG1usap"
        }
      ]
    },
    "basicUsersData": {
      "users": [
        {
          "name": "adam@iacustomer.com",
          "password": "zkv4CnQdwg"
        },
        {
          "name": "audrey@iacustomer.com",
          "password": "9NXFzR7kbI"
        },
        {
          "name": "bob@iacustomer.com",
          "password": "qImE4a2XrV"
        },
        {
          "name": "connie@iacustomer.com",
          "password": "OL4TafPfKx"
        },
        {
          "name": "emma@iacustomer.com",
          "password": "THLKs8fsZS"
        },
        {
          "name": "imran@iacustomer.com",
          "password": "LjjqwFJpv5"
        },
        {
          "name": "rita@iacustomer.com",
          "password": "Wl6SqzhdXkQx"
        },
        {
          "name": "sue@iacustomer.com",
          "password": "m6UbdSc3PA"
        }
      ]
    },
    "password": "otds"
  }
}
```

5.5.3.2 Authentication

Unlike Vault integration, Spring only supports a limited number of authentication methods for Secrets Manager:

- Authentication using environment variables,
- Authentication using system properties, or
- Authentication using profiles.

For on-prem deployment, it does not matter which authentication method is used, as Spring uses an authentication chain. As long as one of the above methods are selected, Secrets Manager will be able to fetch secrets.

Cloud deployment uses the environment variables method to authenticate Secrets Manager. An infrastructure was created to configure the access key ID (environment variable: `AWS_ACCESS_KEY_ID`) and secret access key (environment variable: `AWS_SECRET_ACCESS_KEY`) and set them as k8s secrets, which are then available to each component's container.

5.5.4 Using Gemalto SafeNet KeySecure with OpenText Information Archive

Gemalto SafeNet KeySecure is a security provider that you can use to encrypt data or passwords in OpenText Information Archive. See section 3 “Connecting system components to CipherTrust/KeySecure appliance” in *OpenText Information Archive - Encryption Guide (EARCORE-AGE)* for more information.

5.5.5 Integration with OpenText Key Mediator

OpenText Information Archive encrypts sensitive information using keys. OpenText Information Archive provides an alternative integration with Key Mediator, which provides a mechanism to encrypt/decrypt keys using a master encryption key (MEK). It also provides the ability to either rotate which MEK is used or indicate that the MEK has been compromised. When this occurs, the IA Shell provides a command (`set-mek-alias`) to inform OpenText Information Archive to use the new MEK alias so the system encrypts/decrypts the keys using the new MEK (see section 2.5.2 “set-mek-alias” in *OpenText Information Archive - IA Shell Guide (EARCORE-ARE)* for more information). Refer to the Key Mediator product documentation to learn more about its installation and options.

Note: There is a hierarchy of keys so changing the MEK does not re-encrypt all of your sensitive information.

If Key Mediator is enabled, the following scenarios prevent IA Server from starting:

- If the authentication strategy is set to `API_KEY` and the `apiKey` is not set, the logs will indicate that there was an authentication error.

- If the authentication strategy is set to `OTDS`, and any of the following values are not set, the logs will indicate that there was an authentication error:
  - `otds.clientId`
  - `otds.clientSecret`
  - `otds.host`
  - `otds.port`
  - `otds.ssl`
- If the `mediator.host`, `mediator.port`, or `mediator.ssl` are not set.

  If the initial MEK alias was not ever set.

  For more information, see section 3.19.1.1 “Crypto keystore section” in *OpenText Information Archive - Administration Guide (EARCORE-AGD)*.

OpenText Information Archive provides the ability to switch between using a database keystore and Key Mediator.

Important

Switching between the two strategies requires that all IA Servers are stopped and only one IA Server is started when switching between a database keystore and Key Mediator.

The MEK in Key Mediator must not be marked as destroyed while the keys are used, as you will lose access to all your data and the IA Server will not start. A backup of OpenText Information Archive will not help. The only method to recover lost data is to restore a backup of key mediator to re-enable the MEK alias.

5.6 Setting up a staging environment

After you finish planning your production configuration, you should set up a staging environment that duplicates the configuration that you want for your production environment. A staging environment gives you the ability to test the configuration before deploying it to production. Ideally, you would have at least three environments:

- A design environment for the development of applications, which could be as minimal as a demo configuration on a developer’s computer
- A staging environment for testing
- A live production environment

To set up a staging environment, follow the same instructions for installing a production configuration. When it comes time to upgrade, you should upgrade your staging environment before your production environment and verify that it works as expected. For more information, see section 1.2 “Best practices: working with a

staging environment" in *OpenText Information Archive - Configuration Guide (EARCORE-CGD)*.

Chapter 6

Before running setup

Depending on what you want to configure, some steps should be done prior to running setup to answer some questions. These steps include creating and setting environment variables for the IA Server and IA Web App, and, if required, installing and configuring OTDS and OpenText Intelligent Viewing.

6.1 Setting the memory for OpenText Information Archive components

To set the memory for IA Server:

- Set the environment variable for `IASERVER_OPTS` to the following:

  ```
  -Xms<minimum>m -Xmx<maximum>m
  ```

  where `<minimum>` is the minimum amount of memory in megabytes, and `<maximum>` is the maximum amount of memory in megabytes (for example, `-Xms8192m -Xmx8192m`).

To set the memory for IA Web App:

- Set the environment variable for `IAWEBAPP_OPTS` to the following:

  ```
  -Xms<minimum>m -Xmx<maximum>m
  ```

  where `<minimum>` is the minimum amount of memory in megabytes, and `<maximum>` is the maximum amount of memory in megabytes (for example, `-Xms128m -Xmx4096m`).

6.2 Downloading and installing OTDS

OTDS needs to be installed for any of the following reasons:

- You want to use OTDS. If using, decide whether to use the SSO mode.
- You want to use OpenText Intelligent Viewing (OTIV).
- You want to use Key Mediator and use OTDS to generate tokens to access the service.
- You want to use Content Aviator.

If you are already have OTDS installed in your existing deployment, it is possible to configure your existing OTDS for use by OpenText Information Archive.

If you do already have other OpenText components integrated with OTDS, you can skip this task and go ahead with configuring OTDS for authentication provider mode or configuring OTDS for SSO mode.

Caution

If you deploy IA Web App to Apache Tomcat and use OTDS in SSO mode, using the `infoarchive.gateway.profile.OTDS` profile, then there is a race condition with the startup of IA Web App and OTDS. This is because IA Web App expects that OTDS is already up and serving on the certificate endpoint. IA Web App uses the OTDS certificate to verify the signature of JWT tokens issued by OTDS. The order in which IA Web App and OTDS start is not predictable.

Therefore, you should deploy IA Web App and OTDS to separate instances of Apache Tomcat. Also, you should make sure that OTDS is up and running before you start IA Web App.

Before you begin: Review the OTDS documentation, which contains the installation prerequisites and important configuration information.

To download and install OTDS:

1. Download OTDS from support.opentext.com (https://support.opentext.com).

   For Windows, the installer is a `.msi` file. During the installation, set the JDK and Tomcat locations.

   For Linux, the installer is a `.tar` file. During the installation, set the JDK and Tomcat locations.

2. Once installed, apply the latest OTDS patch, if any.

3. If you want to use OTDS in SSO mode, configure Apache Tomcat to use SSL.

6.2.1 Setting the login format

OTDS provides the ability to specify multiple login formats, one of which is where a domain prefix can be specified. For more information, see section 3.5 “Defining user attributes” in *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.

The OTDS`oTExternalID4` attribute allows you to set a login with the following format:

```
oTExternalID4 = OPENTEXT\franz
```

Note: The use of an NT Domain name prefix (for example, CORP in `CORP > johndoe`) makes sense when the synchronized partition's IDP is Active Directory-based. For LDAP and non-synchronized partitions, use the name of the partition.

6.3 Installing OpenText Intelligent Viewing (OTIV)

> **Important**
> Ensure that OTDS is installed and running first before you install OTIV.

A keystore and truststore are required for OTIV. The alias is only required for the IA Server and it is recommended to store this information outside of the installation folder to facilitate future upgrades.

To install OTIV:

1. Install OTIV, which means that OTDS needs to be installed and running. For help on installing OTIV, refer to the product documentation:

   Once OTIV is installed, you should see the following services running in the Windows Services application:

   - OpenText Asset Service
   - OpenText Configuration Service
   - OpenText Markup Service
   - OpenText Publication Service
   - OpenText Publisher Service
   - OpenText Search Service
   - OpenText Viewer Service

   These values are defined in the `IntelligentViewing_MSI.properties` file for the host and port and they need to be using `https`.

2. Verify that OTIV installed correctly:

   a. Log in to OpenText Directory Services Administration.
   b. On the **License Keys** screen, ensure there is a licence titled `Viewing`.

Chapter 7

Starting OpenText Information Archive for the first time

To start OpenText Information Archive for the first time, the following steps must be completed:

- Use setup to generate configuration.
- Perform manual configuration for certain options.
- Configure load balancing, if required.
- Configure for online help
- Start the PostgreSQL Servers.
- Start the components in the correct order.
- Install the first time applications.

7.1 Using setup to generate configuration

This section describes the steps required to install and configure system components on separate computers that are all on the same network. When you install system components on separate computers, you should consider installing each component as a system service because the operating system can automatically start the service at boot time and try to restart a service if it stops.

If you want to run the install software without answering a series of prompts, you can run the software with parameters that specify the type of installation that you want. To view all of the installation parameters that are supported, run the `<IA_ROOT>/bin/setup.bat` (Windows) or `<IA_ROOT>/bin/setup` (Linux) installation software with the `--help` parameter.

Tip: One of the setup questions asks whether you want to configure OTDS initialization. If you answer yes, you are then asked if you want to generate an OTDS example partition configuration. Depending on your answer, the setup program creates one or two bootstrap YAML files in the `<IA_ROOT_DIR>/config/otds-init` directory, which will be used to create the required resources. For more information, see Using OTDS bootstrapping to create the required configuration.

To start the install software:

1. Extract the OpenText Information Archive ZIP files to the `<IA_ROOT>` directory.

   Tip: You can skip unzipping the `infoarchive-support.zip` file if you do not plan on installing any of the sample applications.

2. Using a command prompt, run the following command in the <IA_ROOT> directory:

   - Windows: install.bat
   - Linux: ./install

   Note: For advanced use cases, you can instead run either the setup or iasetup scripts in the <IA_ROOT>/bin directory. For more information about the setup and iasetup scripts, see About the install software.

7.2 Manual steps after running setup

When making changes to the configuration files directly, if you add a new property, the updated value is not propagated during future upgrades or patch installations into the new version of the configuration file.

7.2.1 Configuring the Content Security Policy

Because the OTIV integration requires access to scripts and stylesheets from OTIV endpoints, update the Content Security Policy (CSP) configuration in the config\iawebapp\application-csp.yml file to allow access.

Consult CSP documentation to learn how to configure this value and what the keywords are:

```
infoarchive:
  webapp:
    security:
      csp:
        enabled: true
        # The following is a minimal setting for the value of "Content-Security-Policy"
header
        # You may further customize it to suite your needs, but leave the following
default
        # in place.
        value: "default-src 'self' data: blob: 'unsafe-inline' 'unsafe-eval'
10.2.3.4:* ;"
```

Update the value to instruct CSP to consume OTIV services. The example above assumes that all of the OTIV services mentioned in the application-infoarchive.gateway.profile.OTIV.yml file are running on 10.2.3.4. List all the unique hostnames/ip addresses for each service if they are different.

```
otiv:
  viewer:
    url: "https://10.2.3.4:3358"
  publication:
    url:"https://10.2.3.4:3356"
  search:
    url:"https://10.2.3.4:3357"
```

7.2.2 Manually configuring Vault

Refer to Populating Vault with secrets for more information.

7.2.3 Manually configuring OpenText Information Archive for OTDS

Refer to either Configuring OTDS for SSO mode or Configuring OTDS for authentication provider mode for more information.

1. Make the following changes to the `config/iaserver/application-otds.yml` file:

   a. Set `OTDS.LOCATION.host` to the location of OTDS (do not use `localhost`).
   b. Set the `OTDS.LOCATION.httpsPort` to the same value that you used when prompted by setup.
   c. Set the `OTDS.LOCATION.port`. Because OTDS supports both protocols, use the appropriate port.
   d. Set the `OTDS.LOCATION.protocol` to how you want to connect to OTDS. Supported values include `http` or `https`.
   e. Specify the correct credentials for `OTDS.username` and `OTDS.password`. The password may need to be encrypted if the configuration indicates as much.
   f. Optional If you plan to use Content Aviator, enter the `OTDS.infoarchive.clients.server.clientSecret` tos the same value that you used when prompted by setup:

```
OTDS:
  LOCATION:
    authUrl: ${OTDS.LOCATION.protocol}://${OTDS.LOCATION.host}:$
{OTDS.LOCATION.port}/otdsws/oauth2/token
    host:
    httpsPort:
    path: /otdsws/rest
    port:
    protocol: http
    url: ${OTDS.LOCATION.protocol}://${OTDS.LOCATION.host}:$
{OTDS.LOCATION.port}${OTDS.LOCATION.path}
  infoarchive:
    clients:
      server:
        clientId: infoarchive.server
        clientSecret:
    resource:
      id: infoarchive
  password:
  username:
security:
  oauth2:
    resource:
      jwt:
        keyValue: NOTUSED
```

   You should not have to change the `OTDS.location.baseUrl`, assuming you set the `protocol`, `host`, and `port` correctly.

2. Verify the following changes to the `config/iaserver/application-https.yml` file:

a. Configure key alias based on your keystore and truststore.

b. Review the keystore and truststore information:

```
server:
 ssl:
  ...
  keyPassword: <password>
  keyStore: C:\IA\iadata\server\keystore.p12
  keyStorePassword: <password>
  keyStoreType: PKCS12
  trustStore: C:\IA\iadata\server\truststore.p12
  trustStorePassword: <password>
  trustStoreType: PKCS12
```

3. Update the OTDS password in the `config/iawebapp/application-infoarchive.gateway.profile.OTDS.yml` file:

```
OTDS:
  infoarchive:
    clients:
      cli:
        timeout: 200000
      gateway:
        clientId: infoarchive.gateway
        clientSecret:
        scope: otds:groups
      iawa:
        clientId: infoarchive.iawa
        clientSecret:
        logoutUrl: ${OTDS.infoarchive.gateway.protocol}://$
{infoarchive.gateway.host}:${infoarchive.gateway.port}$
{infoarchive.gateway.contextPath}logout
        scope: otds:groups
    gateway:
      logoutSuccessUrl:
      protocol: http
    resource:
      id: infoarchive
  location:
    contextPath: /otdsws
    host: localhost
    httpsHost: ${OTDS.location.host}
    httpsPort: 8443
    logoutUrl: https://${OTDS.location.httpsHost}:${OTDS.location.httpsPort}$
{OTDS.location.contextPath}/logout
    path: ${OTDS.location.contextPath}/rest
    port: 8090
    url: http://${OTDS.location.host}:${OTDS.location.port}${OTDS.location.path}
  password:
  urlEncodeBeforeBase64: true
  username: otadmin@otds.admin
```

The `clientSecret` values for the `infoarchive.gateway` and `infoarchive.iawa` are populated by the setup process.

Important

If you updated the 25.2 version of the `spring.security.oauth2.client.registration.infoarchive.iawa` section in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.OTDS.yml` file, you must manually update the section again for the 25.3 version. During development of the 25.3 version of the product, the section was renamed to `spring.security.oauth2.client.registration.infoarchiveIawa`. Due to technical limitations, this particular section does not inherit any manual changes made in 25.2 during an upgrade to 25.3 and the changes need to be redone manually instead.

7.2.4 Using OTDS bootstrapping to create the required configuration

This section only applies to new installations that require OTDS. If you are upgrading a system that uses OTDS, OTDS is already configured and so the following steps should not be performed.

For the integration with OTDS, it is necessary to configure OTDS resources, such as partitions, OAuth2 clients, and groups. The setup program creates one or two bootstrap YAML files to create the required resources. Setup asks various questions that are used to create a bootstrap file that are used to create these resources in OTDS. This bootstrap file only creates the minimum configuration. OTDS reads this file on start up.

Without registering groups and users, however, and associating the appropriate roles with those groups, it is not possible to login to the system using OTDS:

- For testing purposes only, it is possible to request setup to generate a secondary optional bootstrap configuration file to define example groups and users.
- For production scenarios, however, configure OTDS with your own groups and users instead. If you want, use the example groups and users bootstrap file generated by setup as a template to register your own groups and users. There are other methods to configure OTDS with groups and users but are out of scope for the OpenText Information Archive installation process.

  Note: If other methods are in fact used, do not forget to associate the applicable roles with those groups. At the bare minimum, ensure there is at least one user with the Administrator role, which can be used to customize the group-to-role mappings through the IA Web App's **Administration** > **Groups** tab.

If the example groups and users have been deployed, but are no longer needed, disable the corresponding `iacustomer.com` partition to suppress the ability for the example users to log in; alternatively, delete the partition entirely.

Important

The OTDS bootstrap YAML configuration mechanism does not support encrypted passwords so, in case of any security concerns regarding plain text

passwords in the configuration files, the bootstrap files should be removed from the filesystem once OTDS has been successfully started using them to configure OTDS, accordingly.

Complete the following:

1. Stop the Apache Tomcat instance that is hosting OTDS.
2. Modify the `catalina.bat` (for Windows) to add an option for the configuration file:

   ```
   -Dotds.config.file=<IA_ROOT_DIR>/config/otds-init/otds-bootstrap-config-init.yml
   ```

3. Start the Tomcat instance hosting OTDS and let it finish (indicated by a line similar to the following):

   ```
   INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [49797]
   milliseconds
   ```

4. To verify this was done correctly, login using the OTDS Admin client and click **Partitions**. There should be `infoarchive` entry with seven roles, no users or groups.

   Note: If you decided to not configure the example users and groups, you will not be able to login to the system unless you create your own users and groups and map them to the roles.

5. Optional If you did ask to create the example configuration, repeat steps 1-3 but change the file to `otds-bootstrap-config-example.yml` (it will be in the same folder).

   To verify this was done correctly, login using the OTDS Admin client and click **Partitions**. You should see an `iacustomer.com` entry with eight users, seven groups and no roles. To verify that the `sue@iacustomer.com` was created correctly, click `Users * Groups`, find Sue and click **Edit Membership**. Sue should be a member of all seven groups. This is important so that you can install the first-time applications using this user.

7.2.5 Creating groups and users using OTDS Admin

To use OTDS for OpenText Information Archive, it is necessary to configure users that can login and assign them to groups which themselves are assigned to OpenText Information Archive roles.

Originally, OTDS Initializer was used to configure OTDS with the initial setup of those roles as OTDS groups in the `infoarchive` partition using a `ROLE_` prefix and the name matching the actual role name.

Starting with 25.2 for on-prem deployments, however, OTDS is being initialized using the OTDS bootstrap YAML mechanism, which introduced the use of OTDS application roles to represent the OpenText Information Archive roles instead.

These application roles are also created in the `infoarchive` partition, but simply share the same name as OpenText Information Archive roles without the `ROLE_` prefix.

For backwards compatibility reasons, when the legacy `ROLE_` prefixed role groups exist, the group to role mapping in OpenText Information Archive only uses those and ignores the OTDS application roles.

Otherwise, the OTDS application roles are used for the group to role mapping instead.

This is an important distinction because it determines which of these 'roles' the actual groups need to be assigned to.

Additionally, OpenText Information Archive does not expect/support users to be directly assigned to roles and needs users to be exclusively assigned to groups which, in turn, are assigned to roles instead.

OTDS itself does support users to be assigned directly to roles and naturally also to the (legacy) role groups that represent OpenText Information Archive roles, but this can cause unexpected/unsupported behavior for the system and should be avoided.

The new OTDS application roles are created unconditionally to facilitate future migration, but in case the legacy role groups are present, exclusively those should be used to assign regular groups to.

This typically applies to all cloud based deployments, regardless of version and any on-prem deployment that was originally installed using a version of OpenText Information Archive before 25.2.

Only when the legacy role groups do not exist should the OTDS application roles be used to assign groups to.

This typically only applies to on-prem installations that were initially installed using OpenText Information Archive 25.2 or later.

In a future version of OpenText Information Archive, we intend to migrate existing OTDS deployments away from using these legacy role groups and harmonize the group to role mapping for all deployments going forward.

7.2.6 Configuring OTIV using OTDS Admin

To configure OpenText Information Archive to use OTIV:

1. Run the OpenText Information Archive setup program.

   a. If you wish to have the example users and groups configuration, answer yes to `Generate OTDS Initialization Information Archive Example partition configuration`:

   Note: You can always use the OTDS Admin client to add your own partitions, groups, and users.

   b. Answer yes to use `OTDS SSO`. This is required for OTIV integration.

   c. Answer yes to enable `OTIV`. There are manual steps required as setup does not update everything required. For more information, see Manually configuring OpenText Information Archive for Intelligent Viewing (OTIV).

2. Check the IA Web App's `application.yml` file to ensure the system automatically populated the IP address for the Gateway host properly. If it is not entered correctly, the redirect after logging out will not work:

```
infoarchive:
  gateway:
    client:
      ssl:
        key-store: C:/OTIV/IA/iadata/gateway/keystore.p12
        key-store-password: xxxxxx
        key-store-type: PKCS12
        trust-store: C:/OTIV/IA/iadata/gateway/truststore.p12
        trust-store-password: xxxxxxx
        trust-store-type: PKCS12
    contextPath: /
    firewall:
      enabled: false
      host-names:
    host: XX.X.XXX.XXXX
```

3. Ensure `infoarchive.gateway.host` points to the Gateway and is not set to `localhost`.

Further configuration steps are required. For more information, see Manually configuring OpenText Information Archive for Intelligent Viewing (OTIV).

Perform the following after OTIV has been installed:

1. Use OTDS Administration to ensure the OpenText Information Archive partition is created.

   Update to the following permissible scopes and default scopes for the `iv-publisher` OAuth client (`read_any_markups` and `write_any_markups` are added, by default):

   - `otds:groups`
   - `otds:roles`
   - `create_publications`

- `view_any_publication`

Refer to the OTDS Administration documentation to learn how to complete the step above.

2. Add the `iv-publisher` user into a group that has end user privileges. Alternatively, if no group is available, select **Edit Application Roles** and click **Assign to Members** and add `iv-publisher` user.

3. By default, OpenText Intelligent Viewing (OTIV) provides basic functionality. However if you wish to use a full license for certain groups, use the following procedure, although it is recommended associating to your own groups:

   Note: It is possible to access some OTIV functionality without assigning a license.

   a. Access the **User & Groups > Groups** tab. For each group:

      - Select **Allocate to License**.
      - Select `Viewing -iv`.

        Note: If you try to do this for the group multiple times, an error is issued.

   b. To ensure the groups are configured correctly, view the **License Keys > Counters** for `INTELLIGENT_VIEWING`. The expected groups should be displayed.

7.2.7 Manually configuring OpenText Information Archive for Intelligent Viewing (OTIV)

Before completing the following instructions, make sure to update the Content Security Policy (CSP). See Populating Vault with secrets for more information.

In the `application-infoarchive.gateway.profile.OTDS.yml` file, any `password` or `clientSecret` values are expected to be encrypted if the following is set in the IA Web App's `application.yml` file:

```
passwordEncryption:
  enabled: true
```

Ensure that the `clientSecrets` match the values specified when you ran setup.

To manually configure OpenText Information Archive for OTIV:

1. Update the following section in the `config\iawebapp\application-infoarchive.gateway.profile.OTIV.yml` file:

```
otiv:
  publication:
    url: http://localhost:3356
  search:
    url: http://localhost:3357
```

```
  viewer:
    url: http://localhost:3358
```

To this (assuming OTIV is installed on `10.1.2.3`):

```
otiv:
  publication:
    url: https://10.1.2.3:3356
  search:
    url: https://10.1.2.3:3357
  viewer:
    url: https://10.1.2.3:3358
```

Ensure that you update the URL to match where OTIV is installed. In most cases, you will also need to use `SSL` (`https` instead of `http`). Check the settings when you installed OTIV.

Refer to configuring OTDS for authentication provider mode and configuring OTDS for SSO mode for more information.

Note: Because the OTIV integration requires access to scripts and style sheets from OTIV endpoints, the Content Security Policy (CSP) configuration in the `config\iawebapp\application-csp.yml` needs to be updated to allow access.

2. Update the following in the `iawebapp\application-infoarchive.gateway.profile.OTDS.yml` file:

   `OTDS.infoarchive.clients.gateway.clientSecret`
   The setup program prompts you for this value (client secret for `infoarchive.gateway`).

   `OTDS.infoarchive.clients.iawa.clientSecret`
   The setup program prompts you for this value (client secret for `infoarchive.iawa`).

   `OTDS.infoarchive.clients.iawa.logoutUrl`
   This value must use `HTTPS`.

   `OTDS.location.host`
   Host name for where OTDS is installed if you are not using SSL. Change the default `localhost` to the correct IP address, as it will not work correctly if left as `localhost`.

   `OTDS.location.port`
   Only required if connecting to OTDS and you are not using SSL.

   `OTDS.location.url`
   This is the URL that the Gateway uses to connect to OTDS. If you wish to use `https`, update the value to `https://${OTDS.location.host}:${OTDS.location.httpsPort}${OTDS.location.path}`.

   `OTDS.location.httpsHost`
   This is the location of the hostname of where OTDS is located.

   `OTDS.location.httpsPort`
   Only required if connecting to OTDS and you are using SSL.

`OTDS.url`
Default is fine unless you want to use SSL to connect to OTDS. If using SSL to connect to OTDS, you must update the value. An example can be found in the comments stored in the `config-templates` version.

`OTDS.password`
The OTDS password. Passwords may need to be blank (for Vault), encrypted, or be in plain text. If you want to encrypt passwords, setup can do this or, if you change your mind, you can encrypt them later using the `iasetup encrypt-passwords` command. For more information, see Profile-specific configuration templates.

The `OTDS.infoarchive.gateway.logoutSuccessUrl` can be left blank and the `OTDS.username` can be left as the default, unless it was changed when installing OTDS.

Refer to Enabling OTIV integration for more information.

7.2.8 Enabling rules for new installations and upgrades

Because of the security implications surrounding the use of rules, rules are disabled, by default. To use rule based processing, the feature must be enabled for new installations and upgrades. To learn more about rules and how they are used, see section 9.9 “Rules” in *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*.

Three jobs use rules to process archived data:

- Apply Hold Rule to Records job
- Apply Retention Rule to Records job
- Trigger Event Rule job

By default, these jobs are hidden from view in new installations and upgrades. To enable the rule feature and access these jobs, add the following profile (highlighted in **bold** text) in the `application.yml` file for each IA Server:

```
spring:
  cloud:
    config:
      enabled: false
  jmx:
    enabled: true
  profiles:
    group:
      no-background-processing:
      - infoarchive.ias.profile.no.cleanup.rollForward
      - infoarchive.ias.profile.no.cleanup.locks
      - infoarchive.ias.profile.no.backup.structured.libraries
      - infoarchive.ias.profile.no.backup.retention.libraries
      - infoarchive.ias.profile.no.orderItem.handling
      - infoarchive.ias.profile.no.batch.handling
      - infoarchive.ias.profile.no.job.handling
      - infoarchive.ias.profile.no.batchProgressCheckTrigger.handling
      - infoarchive.ias.profile.no.saved.search.rerun
```

```
    include:
       - infoarchive.ias.profile.rules
```

Important

Ensure the correct formatting of the profile is used and consistent across all IA Server instances to avoid errors. If enabled properly, the following message should appear when you start the IA Sever: `The following profiles are active: infoarchive.ias.profile.rules`.

If you upgraded your system and no longer require previously created rules, use the IA Shell to delete them:

1. Set the content to the rule being deleted.
2. Run the `delete` command.

7.2.9 Disabling IA Shell logging

By default, IA Shell logs at `WARN` level and logs what commands the user did as history. It is possible to change both settings via the IA Shell's `application.yml` file to change the log level or disable all history logging.

The `iashell-history.log` maintains a list of all commands run by IA Shell. The system, however, allows you to further limit the amount of logging information. After setup generates the `<IA_ROOT>/config/iashell/application.yml` file, update it to:

- Disable the system from logging every command into the `iashell-history.log` file.
- Update the default log level (for example, to `ERROR`) to further limit the amount of logging information that is recorded.

The `<IA_ROOT>/config-templates/iashell/application.yml` file includes an example of how to make these changes:

```
# Uncomment this to change the root logging level or disable logging of the command in
iashell-history.log
#logging:
#  level:
#    root: ERROR
#    com.emc.ia.cli.services.historylog.TraceableShell: OFF
```

Caution

You must not change anything in the `<IA_ROOT>/config-templates/iashell/application.yml` file. Simply use the information to update the `<IA_ROOT>/config/iashell/application.yml` file.

7.3 Enabling HTTP/2 connections

HTTP/2 (also known as HTTP2 and HTTP/2.0) is a recent revision of the HTTP network protocol. When you enable HTTP/2 for OpenText Information Archive, this protocol can improve the speed of page loading when users access IA Web App, and it can improve the speed of REST API calls. By default, HTTP/2 is not enabled.

Most modern browsers (for example, Google Chrome and Mozilla Firefox) support HTTP/2, but only over connections that use TLS 1.2 or newer. Because of this, if you want these browsers to realize HTTP/2 performance improvements with OpenText Information Archive, you must do the following:

- After you enable HTTP/2 according to the procedure below, you must also enable TLS/SSL. For more information about enabling TLS/SSL, see Protecting data in transit.
- To make an HTTP/2 connection, the browser must use the HTTPS protocol to connect to IA Web App. This means that the URL must begin with `https://`.

REST clients can use HTTP/2 without TLS, using the HTTP protocol. If you only want your REST clients to connect using HTTP/2, you can enable HTTP/2 connections without enabling TLS/SSL.

To enable HTTP/2 connections:

1. In a text editor, open the `<IA_ROOT>/config/iawebapp/application.yml` file.
2. Set the `server.http2.enabled` parameter to `true`.
3. If you want modern browsers to connect to IA Web App using HTTP/2, then set up TLS/SSL.
4. Restart IA Web App.

7.4 Provide the online help on a local help server (Private Help Server)

The online help for this module is delivered using the OpenText Global Help Server (GHS) system, which provides your users with live access to the latest version of the help. If you cannot use the GHS system, for example, if your site does not have internet access, you can install the OpenText Private Help Server (PHS), a local version of the help system that can host your OpenText online help on your organization's network. After the PHS is installed, you can then configure your OpenText module(s) to forward all online help requests to your PHS. For detailed information about installing the PHS, see *OpenText Help System - Private Help Server Administration Guide (OTHS-AGD)*.

Notes

- The Private Help Server can support multiple OpenText modules. If the Private Help Server has already been installed within your organization to

support another OpenText module, you can add additional OpenText module online helps to that installation.

- If you are replacing a previous PHS installation, see section 2.5 "Updating a Private Help Server installation" in *OpenText Help System - Private Help Server Administration Guide (OTHS-AGD)*.
- If the server you want to use for the PHS installation cannot connect to the internet, see section 1.1 "Deploying online help files in an environment without Internet access" in *OpenText Help System - Private Help Server Administration Guide (OTHS-AGD)*.

Once the PHS is installed or upgraded, you can use its Online Help Deployer to download online helps from the GHS system by entering the help deployment codes listed in "Help deployment codes" on page 78. For more information about using the codes, see section 3 "Adding product online help to the Private Help Server" in *OpenText Help System - Private Help Server Administration Guide (OTHS-AGD)*.

Table 7-1: Help deployment codes

Code	Product
EARCORE250400-IGD	OpenText Information Archive CE 25.4

7.4.1 Configuring OpenText Information Archive to use a local help server (Private Help Server)

After you install a Private Help Server and deploy the My Product online help, you must redirect help requests to it.

To redirect help requests:

- Update the `urlPrefix` setting that is found in the `<IA_ROOT>\config\iawebapp\application.yml` file. The `urlPrefix` setting can be for a staging, production or private environment.

```
webapp:
  customization:
    location: file:config/iawebapp/customization/
  enableDateAndNumberLocalization: true
  help:
    urlPrefix: https://docsapi.staging.opentext.com/mapperpi
    version: 220200
  sessionTimeout: 1730
```

7.5 Starting the PostgreSQL Servers

Refer to PostgreSQL documentation for more information. Ensure that all PostgreSQL data nodes are started, including the system and all structured data nodes (for the table applications).

If you are not using an external PostgreSQL Server, start all the OpenText Information Archive instances of PostgreSQL Server by opening a command prompt, going to the `<IA_ROOT>\psql` directory, and running the following command:

- Windows: `bin\pg_ctl.exe -Dconf start`
- Linux: `bin/pg_ctl -Dconf start`

7.6 Configuring runtime options for the IA Server

It is possible to use JAVA_OPTS to specify the following options for the IA Server:

- Changing the amount of memory.
- Changing the folder for the temporary files.

  Tip: You can also specify these memory values in gigabytes (for example, `-Xmx4g`).

- Changing the version of PostgreSQL that the IA Server works with. For more information, see "Running the system with a different version of PostgreSQL than the one included in the distribution" on page 41.

To specify the directory that IA Server uses for java.io.tmpdir:

- Set the environment variable for `IASERVER_OPTS` to the directory that you want to use. For example:

```
IASERVER_OPTS="-Djava.io.tmpdir=/mnt/example/tmp"
```

7.7 Starting the IA Server

Although it is possible to start the IA Server as a service, it is recommended that you start it manually the first time to help with troubleshooting any issues.

To start the IA Server:

1. Using a command prompt, navigate to the `<IA_ROOT>` directory and start the IA Server to check for any errors:

   ```
   bin\iaserver
   ```

2. Optional Install IA Server as a system service. For more information, see Installing system services.

Note: If you change any configuration settings after installing the IA Server, you must restart the IA Server service for the change to take effect.

To verify that IA Server is running:

1. Confirm that the main PostgreSQL server service is running (for example, by using Windows Services or `systemd` tooling or a process listing in Linux).
2. Open a browser and go to `<IA_SERVER_IP_OR_HOSTNAME>:<PORT_NUMBER>/services`. IA Server is running if the browser returns either of the following results:
   - Text that resembles the following:

```
{
  "name" : "Information Archive Home Resource",
  "_links" : {
    "self" : {
      "href" : "http://<IA_SERVER_IP_OR_HOSTNAME>:<PORT_NUMBER>/services"
    },
    "http://<IA_SERVER_IP_OR_HOSTNAME>:<PORT_NUMBER>/product-info"
    }
  }
}
```

   - A request for you to open or save a file of type `application/hal+json`.

Tip: You can also verify if the IA Server running is by looking in the logs `<IA_ROOT>/logs/iaserver` directory.

7.8 Starting IA Web App

Using a command prompt, start IA Web App and check for any errors. For more information, see Starting the components with a command prompt.

Optionally, install IA Web App as a system service if you are not planning to use external tomcat. For more information, see Installing system services.

If you are planning on using external Apache Tomcat, see Deploying IA Web App to Apache Tomcat. Tomcat can be configured to run as a service.

To disable example user accounts:

1. In a text editor, open the `<IA_ROOT>/config/iawebapp/application.yml` file.
2. Add the # symbol to the following line:

```
# - infoarchive.gateway.profile.AUTHENTICATION_IN_MEMORY
```

3. Restart IA Web App.

7.9 Installing the first-time setup applications

Before you use OpenText Information Archive, you must install the following first-time setup applications because they enable key functionality used throughout OpenText Information Archive:

- System
- Audit
- Reports
- Tenant

For more information about all of the OpenText Information Archive applications that are included with OpenText Information Archive, see Connectors for the ETL process.

Before installing any applications, it is necessary to choose a location for the default file system root that is accessible to all IA Server nodes and outside of the `<IA_ROOT>` directory.

In the IA Shell configuration, the `default.properties` file contains a property `fileSystemRoot.path=data/root`. The property should be changed to an absolute path pointing to the chosen location.

When installing the first-time setup applications, this property will be used to create the `defaultFileSystemRoot`.

Changing this location later will require changes through the IA Web App, as well as a copy to the new location of files and directories already created.

It is therefore recommended to ensure the correct location is used immediately.

7.10 Ensuring custom presentations are available for the IA Web App

If a custom presentation was configured for an OpenText Information Archive application, and you did not map a network file system to the OpenText Information Archive application's `iawebapp-resources` directory (for example, `<IA_ROOT>/examples/applications/Tickets/config/application-config/custom-presentations/iawebapp-resources`), then copy the contents of the OpenText Information Archive application's `iawebapp-resources` directory into the `<IA_ROOT>/config/iawebapp/customization/custompresentations` directory on each of the computers that hosts IA Web App.

Note: If you are using external Tomcat, the procedure is different. For more information, see Deploying IA Web App to Apache Tomcat.

If you are installing new applications that have custom presentations, repeat this procedure to ensure that all IA Web App instances have access to the custom presentation images.

7.11 Installing system services

If you install an OpenText Information Archive component as a system service, and then restart the computer, the component starts automatically. For more information, including limitations to the scripts for installing services, see System services.

Before you begin:

- On Linux, verify that you have, or can acquire, root privileges.
- On Windows, verify that you have, or can acquire, administrative privileges.

To install system services:

1. Stop all OpenText Information Archive components, including the ones that you started from command prompts (IA Server, and IA Web App).
2. In a command prompt, run the `install_services.bat` (Windows) or `install_services` (Linux) installation script located in the `<IA_ROOT>/bin` directory.
3. Follow the prompts and choose to install services for the OpenText Information Archive components that you want. This registers the services with the operating system so that it can start them from the appropriate directory.

7.12 Integrating the Content Aviator cloud bridge

For content systems that use Content Aviator outside of managed private cloud environments (Google Cloud Platform (GCP, Azure, or Amazon Web Services) (AWS), a hybrid deployment is required. In this model, Content Aviator runs as a dedicated single-tenant SaaS instance within a managed private cloud project (GCP, Azure, or AWS) and uses cloud bridge components to connect with your content system. This section explains how to connect your content system to Content Aviator using the hybrid deployment option. Complete the following procedure to set up the connection.

7.12.1 Prerequisites

When deploying OpenText Information Archive over OTDS, the following details need to be provided after OpenText Information Archive is successfully deployed. The values for proxy credentials, chat URL, embedding URL, and web socket URL are provided by OpenText:

OTDS URL

For example:

```
http://<ip>:<port>/otdsws
```

OpenText Information Archive service URL

```
http://<ip>:8765/restapi
```

Once the above URLs are provided, the following additional details are required:

Proxy authentication credentials

For example:

```
"username": "testUser",
"password": "testPass"
```

Chat service URL

For example:

```
csai-chat-svc.cluster1.otl-eng-csd-info-archive.otxlab.net
```

Embedding Service URL

For example:

```
csai-embed-svc.cluster1.otl-eng-csd-info-archive.otxlab.net
```

Web socket URL

For example:

```
saas-proxy-service.cluster1.otl-eng-csd-info-archive.otxlab.net
```

7.12.2 Configuring SaaS proxy client

You must use Java 17 or above to complete the following procedure.

1. Download the JAR file from this external URL (https://mimage.opentext.com/support/ecm/cloudbridge/saas-proxy-client-1.0.0.jar).
2. Use either of the following methods to run the `saas-proxy-client` JAR file:
   a. To use the configuration file, navigate to the build directory:

   ```
   cd [OTHOME]\bin
   ```

   Create a `config.properties` file inside the `<OTHOME>\bin` directory. This file must contain the required configuration. The values that need to be updated appear in **bold** in the following example:

```
## Saas-proxy-service config.
websocket.endpoint=wss://saas-proxy-service.cluster1.otl-eng-csd-info-
archive.otxlab.net
websocket.auth=testUser\:testPass
websocket.connections.per.client=5

## server urls config
servers.otds=https://<ip>:8090/otdsws
servers.ias=https://<ip>:8765/restapi

## Whitelisting endpoints. Static for content system
callbacks.keys=otds,ias
callbacks.otds.headers=content-type
callbacks.otds.method=get
callbacks.otds.path=/oauth2/jwks
callbacks.otds.serverRef=otds
callbacks.ias.headers=authorization
callbacks.ias.method=head
callbacks.ias.path=/systemdata/applications/.*/search-results/.*
callbacks.ias.serverRef=ias
response.headers.whitelist=content-type
response.whitelist=headers,status,data
```

Launch the SaaS proxy client using the configuration file:

```
java -jar saas-proxy-client-1.0.0.jar
```

b. To generate a configuration window from the properties file, run the JAR using the following command:

```
java -jar saas-proxy-client-1.0.0.jar ui
```

Running the JAR file with the flag UI generates a configuration window where you can modify the needed configuration to run the `SaaS-proxy-client`.

Next, complete the following procedure to enable Content Aviator by updating some global settings.

1. Log into the IA Web App as an Administrator.
2. Navigate to **Administration > Global Settings** and configure the following:
   a. Set `aviator.enabled` to `true` and click **Update**.
   b. Set `aviator.embeddings.service.url` to the desired URL and click **Update**. For example:

```
sai-chat-svc.cluster1.otl-eng-csd-info-archive.otxlab.net
```

   c. Set `aviator.chat.service.url` to the desired URL and click **Update**. For example:

```
csai-embed-svc.cluster1.otl-eng-csd-info-archive.otxlab.net
```

Chapter 8

Install troubleshooting

The command prompt window displays installation and upgrade status messages, including errors that may occur during the installation or upgrade. You can also find status messages in the log files in the `<IA_ROOT>/logs/iasetup` directory. You should look at the log files even after a seemingly successful installation or upgrade.

Note: You can locate errors by searching for occurrences of the word `WARN` or `ERROR` in the log files.

8.1 Running the install software

Issue: **Unable to run the install software.**

Check that JDK is installed properly by doing the following:

1. Make sure that the `JAVA_HOME` environment variable is set properly for the service account, and the version of JDK matches the version from *OpenText Information Archive Release Notes* on support.opentext.com (https://support.opentext.com/).
2. On Linux, run the `alternatives --config java` command. It should return a single result.
3. Make sure that there is only one version of JDK installed.

8.2 Running the service for an OpenText Information Archive component after installation

Issue: **Unable to run the service for an OpenText Information Archive component.**

For Windows, do the following:

1. Look at the installation template and transcript, if you chose to generate them when finalizing the install, and make sure that the values are what you would expect.
2. Verify that the ports that you have specified for OpenText Information Archive services are not already in use by other services.

For Linux, do the following:

1. Generate the service log by running the following commands:

```
systemctl status <SERVICE_NAME>.service
```

where `<SERVICE_NAME>` is the name of the service that you are experiencing issues with.

```
journalctl -xe
```

If something goes wrong, you can look through the service log yourself, or share the service log with OpenText Customer Support.

2. Verify which user installed the component and service, which user is running the service, and verify that the appropriate environment variables are set for the user as follows:

   - When you install an OpenText Information Archive component, you must not run the install software with the root account.
   - When you install an OpenText Information Archive service, you must not run the `install_services` script with the root account. You will be asked to grant these privileges to the script. The user account that runs the services (in other words, the service account) should also not be the root account.

     If you did install a component or a service with the root account, then uninstall the component or service as the original user, log in as the appropriate user, and then reinstall the component or service.
   - If you run services with the `systemctl` utility, you should check that the appropriate environment variables can be used by the utility. To dump the `systemd` manager environment block, run the following command:

     ```
     systemctl show-environment
     ```

     This list should include the `JAVA_HOME`, `PATH`, and `IASERVER_OPTS` variables. If there are any missing variables, then you must import them using the following command:

     ```
     systemctl import-environment
     ```
   - You can also run the following command:

     ```
     alternatives --install /usr/bin/java java /<JAVA_PATH>/bin/java
     1
     ```

     This command creates a symbolic link in `/usr/bin`, and by default, `/usr/bin` is available in the PATH environment variable of `systemctl`. This means that the `java` command will be available for `systemtcl`. If you do not install `alternatives` for Java, then you must import the PATH variable for `systemctl`, using the following command:

     ```
     systemctl import-environment
     ```

3. If the component does not start after a new installation, do the following:

   - The installation process for OpenText Information Archive version 4.3 and later can create automatically generated passwords and secret tokens for OpenText Information Archive components. These passwords and tokens are automatically set in several configuration files. But a component will not function correctly if you do not use the same password or token for the component on each computer that you install an OpenText Information Archive component on.

4. If the service for IA Server or IA Web App does not start, and you encrypted component passwords and secret tokens in configuration files, do the following:

   - Check whether you enabled `noPromptStartup` and whether the `secretStore.uber` and `creds` files exist. If you are prompted for a keystore password, specify the password. For more information, see section 7.2.3 "Using the password encryption utility to manually encrypt passwords and tokens" in *OpenText Information Archive - Encryption Guide (EARCORE-AGE)*.

8.3 Installing an OpenText Information Archive application and running IA Shell commands

Issue: **Unable to install an OpenText Information Archive application.**

- Make sure that the PostgreSQL database clusters, IA Server, and IA Web App are running.
- Try to log in to OpenText Information Archive. If you cannot log in, then make sure that IA Server and IA Web App are running using compatible profiles. For example, the following parameters in the components' configuration files must match:
  - The Gateway client secret key
  - If you are using OTDS SSO, the profiles in the components' `application.yml` files

  Also, look for issues with the `application.yml` file, such as spacing problems.
- Make sure that all the configuration files for your application that are in YAML format have had their parameters specified correctly. For more information, see Specifying parameters in configuration files.
- If when you try to install an OpenText Information Archive application, you see the error message `Illegal character in path`, the configuration for the applications was not done. You can confirm this by looking at the installation transcript, if you chose to generate it when finalizing the install.

  To resolve the issue, do the following:

  1. In a command prompt, in the `<IA_ROOT>` directory, run the following command:

Windows	`\bin\iasetup apply --config.review=true`
Linux	`<IA_ROOT>/bin/iasetup apply --config.review=true`

     If you used a different name when you saved the setup configuration as a template, in the command above, substitute the name that you specified for `apply`.

2. Review the questions and when prompted with `Configure Applications: (Y/n)`, type `y`.

- If you are unable to install the example applications in OpenText Information Archive, do the following:

  1. Make sure that the IA Server, IA Web App, and PostgreSQL servers are running.
  2. Do the following:
     a. In a text editor, open the `<IA_ROOT>/config/iashell/application.yml` file.
  3. Verify that the configured `fileSystemRoot` path refers to a directory that can be written to. Especially in case of more than a single IA Server node, this directory needs to be accessible to all IA Server nodes or unpredictable behavior will cause accessibility problems.
  4. Check for the following password mismatches:
     a. *User accounts*:
        – In the `<IA_ROOT>/config/iashell/application.yml` file, if the `connection.userName` and `connection.userPassword` properties are specified, they must be correct. The password can be encrypted based on other settings.
        – If you are using example user accounts, they must match the credentials for one of the user accounts that is included in all of the roles, as specified in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.AUTHENTICATION_IN_MEMORY.properties` file (for example, `sue@iacustomer.com`).
        – If you are using an external authentication server, check that the `connection.userName` and `connection.userPassword` properties match the appropriate credentials.
     b. *Database passwords*:
        – In the `<IA_ROOT>/config/iashell/default.properties` file, check that the `rdbDataNode` and `rdbDatabase` username and password values are correct.
     c. *Keystore passwords*:
        – If you configured TLS/SSL, in the `<IA_ROOT>/config/iashell/application-https.yml` file, check that the `ssl.trustStorePassword` and `ssl.keyStorePassword` parameters are correct.

Issue: **When you try to install an OpenText Information Archive application or run an IA Shell command, you get the error message `Command not found`.**

Make sure that you did not misspell the command in IA Shell or an OpenText Information Archive application's `install.iashell` file.

Note: When running IA Shell commands, as soon as a command fails, processing stops.

For more information about IA Shell commands, including how to get a list of available commands, see section 2 "IA Shell commands" in *OpenText Information Archive - IA Shell Guide (EARCORE-ARE)*.

8.4 Encrypting passwords during installation

Issue: **Password encryption is enabled for one or more components, but disabled for others.**

Passwords will be decrypted for those components where it was enabled, but interpreted as clear text (unencrypted) for those where it was not. The install software can only enable or disable password encryption for all components. Whether or not password encryption will be enabled depends on which components had password encryption enabled and the order in which the install software processes the configuration.

- You can forcibly enable password encryption by running `iasetup --passwordEncryption.enabled=true`.
- You can also encrypt all passwords after install or upgrade by running `iasetup encrypt-passwords`.

Issue: **Password encryption uses different settings across components.**

Passwords will be decrypted according to local component settings, but the install software picks one of the password encryption configurations to encrypt passwords in the generated configuration files.

Issue: **Not all passwords are encrypted because not all configuration files are relevant, given a specific configuration.**

The install software logs a warning about each password that could not be decrypted. In this case, only clear text passwords should be mentioned, in which case there is no problem.

Issue: **Some passwords are encrypted using different password encryption settings than the local configuration for the component.**

The install software logs a warning about each password that could not be decrypted. In this case, encrypted passwords are mentioned, and this is a problem that needs to be investigated before proceeding.

Issue: **The password encryption configuration is inconsistent. For example, the configuration points to a nonexistent keystore, or specifies an invalid keystore password.**

The install software fails to decrypt passwords. You need to correct the configuration before retrying the upgrade or update.

Issue: **Password encryption might have been configured to not prompt for passwords during startup, but the persisted password does not match the password for the keystore being used.**

The install software detects this and asks for the correct password. If this fails and decryption is not working, investigate the problem before proceeding.

8.5 Registering a data node through IA Web App

Issue: **Unable to register a data node through the IA Web App.**

Ensure that the database cluster is running and double-check the connection string. Note that the data node user must have already been created in PostgreSQL.

8.6 Cannot take a backup of a table database or take a table application offline

Issue: **You cannot take a backup of a table database or take a table application offline.**

There may be a mismatch between the IA Server and the data node containing the database that is being backed up (taking an application offline may take a backup). This is applicable for OpenText Information Archive versions 22.2 and onward.

You may not be able to take a table database offline if someone recently ran a table search. Wait a few minutes and try again. For more information, see Running the system with a different version of PostgreSQL than the one included in the distribution.

8.7 Cannot start the PostgreSQL server

Issue: **You cannot start the PostgreSQL server.**

The following assumes you are trying to start the PostgreSQL server that was created via setup:

1. Determine if it is already running by reviewing the processes. Since PostgreSQL typically starts multiple processes, use the `pg_ctl -Dconf stop` command from the previous version.
2. Check the `postgresql.conf` file and ensure that the directory can be found. It is recommended to not use a relative patch for the `data_directory`.
3. Ensure that the setup program ran correctly if data node creation was supposed to be done by the OpenText Information Archive setup program.

8.8 Cannot start IA Server

Issue: **You cannot start the IA Server after configuring PostgreSQL SSL.**

Setting up IA Server and the IA Web App for SSL is completely independent of PostgreSQL SSL, and can be configured differently for system versus structured data:

1. Check which `sslmode` they are using. Both verify options require certificates, the difference is whether the host is also checked.
2. Ensure that `hostssl` and `cert` are specified in `pg_hba.conf`; otherwise, SSL might not be used.
3. If using Windows, the certificate:
   - Must be in DER format, and
   - Must not be encrypted.
4. When configuring the certificate for the server, the CN must match what is specified during the setup questions.

If the client and/or server setup is not correct, you will receive one of the following `SqlExceptions` when attempting to connect to the PostgreSQL server:

`FATAL: certificate authentication failed for user "postgres"`
The given client certificate has a wrong CN (which do not match `postgres`). If using different accounts, you need certificates for each account.

`FATAL: connection requires a valid client certificate`
The user is configured at PostgreSQL to only authenticate via a certificate, but no (readable) certificate is given.

`FATAL: password authentication failed for user "postgres"`
The user is configured at PostgreSQL to provide username and password (at least as fallback), but the password is wrong.

`SSL error: Received fatal alert: decrypt_error`
The given client certificate and the `client.key` do not match.

`The server requested password-based authentication, but no password was provided.`
The user is configured at PostgreSQL to provide username and password (at least as fallback), but no password was provided.

Issue: **Cannot start IA Server on clean install when configuring Key Mediator.**

1. Refer to the console and server logs, which should explain the issue. Common issues include not setting mandatory parameters. Check hosts and ports. Note that usually `https` should be `true` for key mediator and OTDS.
2. Verify that Key Mediator is running and check credentials.

3. Attempting to configure Key Mediator is not supported for existing customers. Plan an upgrade path in later releases.
4. If having trouble with OTDS, it is recommended to use the API Key instead. Once API Key is working, troubleshoot the issue with OTDS.
5. If you are getting an exception like the following, it is because the certification could not be validated. Some options are to not use SSL or check how OTDS was configured for Key Mediator:

```
Caused by: javax.net.ssl.SSLHandshakeException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target
    at java.base/sun.security.ssl.Alert.createSSLException(Alert.java:131)
    at java.base/sun.security.ssl.TransportContext.fatal(TransportContext.java:
353)
    at java.base/sun.security.ssl.TransportContext.fatal(TransportContext.java:
296)
    at java.base/sun.security.ssl.TransportContext.fatal(TransportContext.java:
291)
    at java.base/sun.security.ssl.CertificateMessage
$T13CertificateConsumer.checkServerCerts(CertificateMessage.java:1357)
    at java.base/sun.security.ssl.CertificateMessage
$T13CertificateConsumer.onConsumeCertificate(CertificateMessage.java:1232)
    at java.base/sun.security.ssl.CertificateMessage
$T13CertificateConsumer.consume(CertificateMessage.java:1175)
    at java.base/sun.security.ssl.SSLHandshake.consume(SSLHandshake.java:392)
    at java.base/
sun.security.ssl.HandshakeContext.dispatch(HandshakeContext.java:443)
    at java.base/
sun.security.ssl.HandshakeContext.dispatch(HandshakeContext.java:421)
    at java.base/
sun.security.ssl.TransportContext.dispatch(TransportContext.java:183)
    at java.base/sun.security.ssl.SSLTransport.decode(SSLTransport.java:172)
    at java.base/sun.security.ssl.SSLSocketImpl.decode(SSLSocketImpl.java:1507)
    at java.base/
sun.security.ssl.SSLSocketImpl.readHandshakeRecord(SSLSocketImpl.java:1417)
    at java.base/
sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:456)
    at java.base/
sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:427)
    at java.base/
sun.net.www.protocol.https.HttpsClient.afterConnect(HttpsClient.java:572)
    at java.base/
sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractD
elegateHttpsURLConnection.java:201)
    at java.base/
sun.net.www.protocol.https.HttpsURLConnectionImpl.connect(HttpsURLConnectionImp
l.java:168)
    at
org.springframework.http.client.SimpleBufferingClientHttpRequest.executeInterna
l(SimpleBufferingClientHttpRequest.java:76)
    at
org.springframework.http.client.AbstractBufferingClientHttpRequest.executeInter
nal(AbstractBufferingClientHttpRequest.java:48)
    at
org.springframework.http.client.AbstractClientHttpRequest.execute(AbstractClien
tHttpRequest.java:66)
    at
com.emc.ia.crypt.impl.services.KeyMediatorServiceImpl.transform(KeyMediatorServ
iceImpl.java:124)
```

Issue: **You cannot start IA Server after Key Mediator has been configured.**

- If `crypto.keyStore.control.type` is set to FILE and `crypto.keyStore.control.migrate` is set true, the server will not start. Do not set migrate to true for a new installation.

- If `crypto.keyStore.control.type` is set to `KEY_MEDIATOR`, the server will not start if any of the following values under `crypto.keyStore.mediator` are not set:
  - `host`
  - `initialMekAlias`
  - `port`
  - `authenticationStrategy` (must be either `API_KEY` or `OTDS`)
- If `crypto.keyStore.control.type` is set to `KEY_MEDIATOR` and `crypto.keyStore.mediator.authenticationStrategy` is `API_KEY`, the server will not start if `crypto.keyStore.mediator.apiKey` is not set
- If `crypto.keyStore.control.type` is set to `KEY_MEDIATOR` and `crypto.keyStore.mediator.authenticationStrategy` is `OTDS`, the server will not start unless all the values in `crypto.keyStore.mediator.otds` are set
- The server will not start if either OTDS or Key Mediator are unavailable if configured to use them (configuring OTDS for key mediator is done by setting `crypto.keyStore.mediator.authenticationStrategy` to `OTDS`)

Issue: **Cannot start IA Server when configuring Vault.**

1. Verify that the `application.yml` settings for the configuration for the Vault are correct.
2. Verify that you imported the JSON for the keys to be managed by Vault.
3. If Vault is configured, the sensitive information in configuration files is blank.

8.9 Troubleshooting OTDS issues

Ensure that all the configuration steps are performed (for example, proper spring profiles for both server and gateway).

Refer to either Configuring OTDS for SSO mode or Configuring OTDS for authentication provider mode for more information.

If you are still experiencing issues, if you are using OTDS SSO, ensure that the OTDS host is set to the IP address and not `localhost`, unless everything including the browser is on the same machine. You can also check that the correct profile is enabled in the `iawebapp\config\application.yml` file.

If you are still experiencing issues while using OTDS SSO, ensure that the OTDS host is set to the IP address and not `localhost`, unless everything including the browser is on the same machine. The OTDS host is defined in the OTDS profile under `<INFOARCHIVE_ROOT>/config/iawebapp`.

Ensure that the `truststore` in the `<INFOARCHIVE_ROOT>/config/iawebapp` directory is in place and has the Tomcat certificate.

If you still experience issues, see section 25 “Troubleshooting” in *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.

Issue: **After configuring OTDS, the following error is issued when clicking the OTDS button: `No redirect URI registered`.**

Connect to OTDS as the user `otadmin@otds.admin`, locate the `infoarchive.iawa` OAuth client and navigate to the **OAuth Clients > Actions > properties> Redirect URLs** tab.

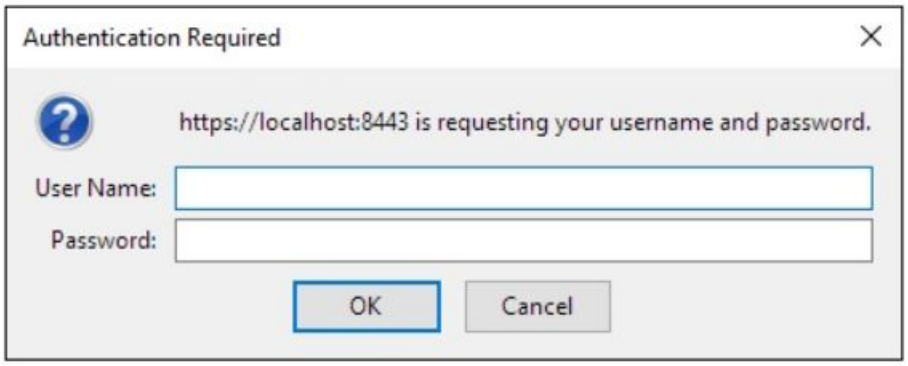
Click the **Add** button. Add the value of `preEstablishedRedirectUri` and click **Save**. Also check if you have a load balancer set up for IA Web App and whether that affects the URI that you specify.

For more information on Creating an OAuth Client, refer to the OTDS documentation.

If you are using external Tomcat, check the port and ensure the URI includes the context (default is `infoarchive-webapp`). For more information, see Deploying IA Web App to Apache Tomcat.

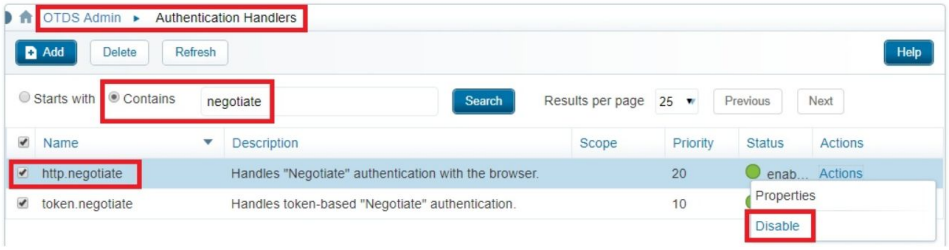
Issue: **Unexpected pop-up is displayed when trying to log into OTDS Admin:**

Depending on the configuration of OTDS and your browser, you may see the following dialog prior to seeing the OTDS login dialog:



Simply **Cancel** the dialog and the OTDS login dialog will be shown. This happens because of the `http.negotiate` authentication handler is enabled in OTDS. You can disable that in the OTDS Admin web application:

1. In the OTDS Admin web application, go to the **Authentication Handlers** page.
2. In the **http.negotiate** row, click **Actions > Disable**.



You need to leave the http.negotiate Authentication handler as enabled, however, if you are using the Kerberos > NTLM > Windows desktop authentication integration with OTDS. This will also require changes to the browser configuration so that the Authentication Required dialog is not shown. For more information on single sign on issues, refer to the OTDS documentation.

Issue: **After configuring OTDS and attempting to log into OpenText Information Archive, a blank screen is displayed.**

Close the browser or disconnect from OTDS. Log back into OpenText Information Archive as a user with the proper rights.

Issue: **When starting up the server, an exception is shown in the console that could not login to OTDS.**

Check your credentials in the IA Server's `application-otds` file. This issue will not prevent the server from starting up. If you try to change the group permissions, however, the changes will not be pushed to OTDS. This means that if you do correct the credentials any changes that were made will be reset to the values that OTDS has.

8.10 Issues when OTIV is enabled

Issue: **OTIV is not working when it has been enabled.**

The Content Security Policy (CSP) is blocking access. The CSP configuration must be updated. See Manually configuring OpenText Information Archive for OTIV for more information.

Issue: **Experiencing issues when running searches when OTIV is enabled.**

Ensure the configuration files have been updated. For more information, refer to Manually configuring OpenText Information Archive OTIV profiles.

Chapter 9

Optional security configuration

9.1 Setting up load balancing

In a production configuration with multiple instances of the same system component, you can set up a server-side load balancer to listen on the port where clients connect to the system component. The load balancer then forwards the request to one of the instances, distributing the load.

Note: In a setup with multiple IA Web App servers, where each of them communicates with its own OTDS, the load balancer is required to direct connections of the session to the same IA Web App server that was used by the session for logging in. The reason for it is that access and refresh tokens issued after a successful login can be validated only by the issuing IA Web App server.

There is no generic way of configuring load balancers for it. For example, F5 requires enabling session persistence profile and NGINX uses `ip_hash` property. Refer to your proxy software documentation for more information.

There are many different load balancers available, with different feature sets, so for more information about your load balancer, consult the documentation that came with it.

You can configure the load balancer in HTTPS mode to use TLS/SSL, with or without also configuring system components to use TLS/SSL. For more information about setting up TLS/SSL for system components, see Protecting data in transit.

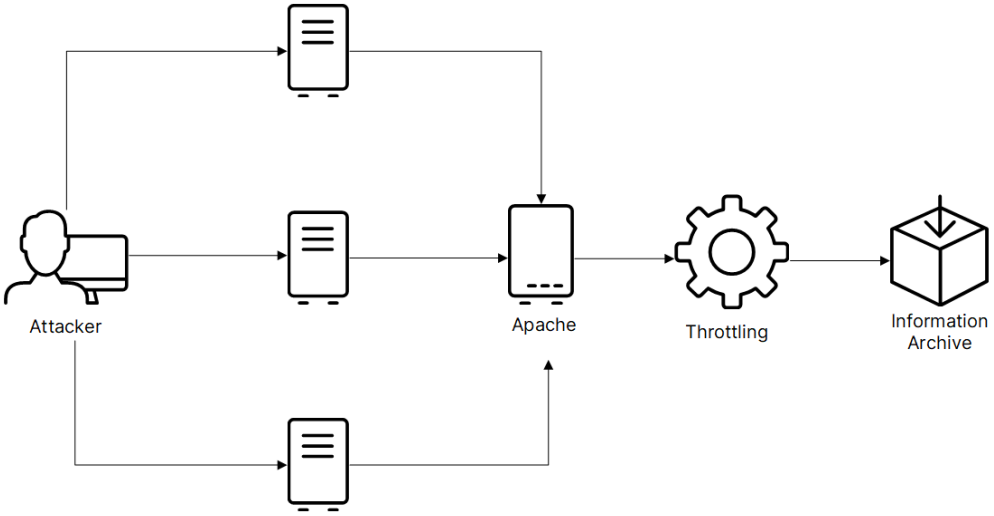
To learn about setting up load balancing, refer to the respective load balancer documentation.

9.2 Throttling access to OpenText Information Archive with Apache HTTP Server

A distributed denial of service (DDoS) attack consists of issuing a high number of requests, sometimes from multiple locations. To help prevent this possible attack from affecting OpenText Information Archive, you can use an external proxy such as Apache HTTP Server to throttle access to OpenText Information Archive. Since this is an external integration, you can choose whatever proxy you prefer for this option. Also, your organization might be using forward-facing proxies already with other enterprise software for the same reason.

This functionality has been tested with Apache HTTP Server and the `mod_evasive` module. Other modules for Apache HTTP Server are also available.

In a typical DDoS attack, an attacker uses several computers to launch the attack, and the computers start flooding the proxy with requests. With Apache HTTP Server throttling the requests, it helps prevent the system from becoming vulnerable to the attack.



In the following simple configuration, any request to the `/services` endpoint gets forwarded to the OpenText Information Archive REST layer. With the configuration of the `DOSPageCount`, `DOSPageInterval`, and other parameters, Apache HTTP Server monitors this traffic. As long as the requests are coming in under the threshold, each request is forwarded. Any request above the threshold is immediately blocked and Apache HTTP Server responds to it with a 403 Forbidden error.

The behavior and response may vary between servers and/or modules doing the monitoring, but the end effect is usually the same, that OpenText Information Archive is protected from a DDoS attack.

To throttle access to OpenText Information Archive with Apache HTTP Server:

1. Install Apache HTTP Server.
2. Deploy the `mod_evasive` module to Apache HTTP Server and edit the `httpd.conf` file to configure it. To load the module, add the following line:

   ```
   LoadModule evasive2_module modules/mod_evasive2.so
   ```

3. Configure Apache HTTP Server as a forwarding proxy to perform a test. You can do this in several ways, including by uncommenting the following three modules in the `httpd.conf` file:

   - `proxy`
   - `proxy_http`
   - `proxy_connect`

4. The following is an example of the default configuration for the mod_evasive proxy in the `httpd.conf` file:

```
<IfModule evasive2_module>
DOSHashTableSize 3097
DOSPageCount 2
DOSSiteCount 50
DOSPageInterval 1
DOSSiteInterval 1
DOSBlockingPeriod 10
</IfModule>
```

5. Configure the forwarding proxy. The following example shows a basic configuration in the `httpd.conf` file for forwarding any request to the `/services` endpoint on Apache HTTP Server to IA Server:

```
<IfModule mod_proxy.c>
ProxyRequests On
ProxyVia On
</IfModule>

ProxyPass /services http://localhost:8765/services
```

9.3 Handling headers used for forwarding HTTP requests

An HTTP header injection attack might consist of an `X-Forwarded-Host`, `Host`, or `Forwarded` header set to a value that the attacker can use to create a vulnerability in the system. This header is typically used in environments where there are one or more proxies in the system. In OpenText Information Archive, the Spring Framework uses the value of the header to rewrite links in the response, which might introduce vulnerabilities.

9.3.1 Handling X-Forwarded-Host and host headers

In a typical scenario without any proxy, when you issue a GET call to the home service for the IA Server, this is a typical response (partial for clarity), assuming the system is deployed on `localhost port 8765`:

```
"http://identifiers.emc.com/product-info": {
    "href": "http://localhost:8765/product-info"
},
"http://identifiers.emc.com/tenants": {
    "href": "http://localhost:8765/systemdata/tenants"
},
"http://identifiers.emc.com/tenant": {
    "href": "http://localhost:8765/systemdata/tenants/3a29310b-5342-441b-990e"
},
"http://identifiers.emc.com/managed-items-restore": {
    "href": "http://localhost:8765/systemdata/managed-items/restore"
},
"http://identifiers.emc.com/managed-items-backup": {
    "href": "http://localhost:8765/systemdata/managed-items/backup"
}
```

If the same request contains an `X-Forwarded-Host` header value that is set, for example, to the value `attacker_host:8888`, the response is as follows:

```
"http://identifiers.emc.com/product-info": {
        "href": "http://attacker_host:8888/product-info"
    },
    "http://identifiers.emc.com/tenants": {
        "href": "http://attacker_host:8888/systemdata/tenants"
    },
    "http://identifiers.emc.com/tenant": {
        "href": "http://attacker_host:8888/systemdata/tenants/
3a29310b-5342-441b-990e"
    },
    "http://identifiers.emc.com/managed-items-restore": {
        "href": "http://attacker_host:8888/systemdata/managed-items/restore"
    },
    "http://identifiers.emc.com/managed-items-backup": {
        "href": "http://attacker_host:8888/systemdata/managed-items/backup"
    }
```

This is an issue you will probably want to address. It would be best for your deployment to be flexible, but also able to distinguish between host values that are good or bad.

One way to achieve this is to create a whitelist of acceptable host values, either by IP addresses or host names, and then only allow requests that contain values from the whitelist. If any request contains an unapproved value, the request fails with the client-side error `400 Bad Request Error`.

To specify a range of acceptable values for the whitelist, use the `server.tomcat.remote-ip.internal-proxies` property in the `<IA_ROOT>/config/iaserver/application.yml` configuration file. The first example allows forwarding only for the host name `www.legitimate.name`:

```
server:
  tomcat:
    remote-ip:
      internal-proxies: www.legitimate.name
```

You can then go a step further by using a regular expression. The following examples show providing a whitelist using simple regular expressions:

1. Using more than one legitimate value allows for load balancing:

   ```
   server:
     tomcat:
       remote-ip:
         internal-proxies:(serverA)|(serverB)
   ```

2. You can allow for either specific host names or a range of IP addresses. The following regular expression allows an `X-Forwarded-Host` header with either a value of `serverA` or with any IP address matching `192.168.*.*`:

   ```
   server:
     tomcat:
       remote-ip:
         internal-proxies: (serverA)|(192\.168\.\d{1,3}\.\d{1,3})
   ```

3. You can match against a host name starting with the value `somedomain` or a range of IP addresses:

```
server:
  tomcat:
    remote-ip:
      internal-proxies: (somedomain.*)|(192\.168\.\d{1,3}\.\d{1,3})
```

4. You can match host names and IP addresses with an optional port:

```
server:
  tomcat:
    remote-ip:
      internal-proxies: (somedomain.*)(:[0-9]+)?|(192\.168\.\d{1,3}\.
\d{1,3})(:[0-9]+)(:[0-9]+)?
```

To accomplish this, you can define a filter and add a predicate for the regular expression's pattern matcher.

This new property is optional and not set to any value by default.

Note: The new filter behavior will only be applicable to requests that set an `X-Forwarded-Host` and/or `Host` header value.

9.3.2 Handling forwarded headers

A `Forwarded` header has a more complex structure. This type of header is more generic and can contain multiple values. For example:

```
Forwarded: for=192.0.2.60, 192.0.2.61;proto=http;by=203.0.113.43
```

This header has several constructs. It consists of the following:

- Elements, with a semicolon (;) as a separator
- Key-value pairs
- Values, which can be either single or HTTP lists

The elements can contain several types that are relevant to validating host names:

- `for`
- `host`
- `by`

Each of these types can carry host information. Typically, the `host` element contains the original `Host` header values, which might have been rewritten by a proxy. This is like the `for` header, which might also contain additional proxies in the chain represented as an `http-list` element. This is somewhat similar to the `by` element.

For the simplicity of configuration and validation, OpenText Information Archive examines all these headers, but one single value is validated based on the following precedence:

1. If the element `host` is present, then validate that value against the whitelist.
2. If the element `host` is not present, but the element `for` is present, then validate that value against the whitelist.

3. If neither the `host` nor `for` element is present, but the `by` element is present, then validate that value against the whitelist.

Because some of these elements might contain values as `http-list`, OpenText Information Archive examines the first element from that list, which typically contains the original `host` value, others being proxies in the chain.

9.4 Additional Gateway security

There is an additional mechanism that can be used to block `X-Forwarded-Host` and `Forwarded` headers from passing through. It is on the Gateway level, behind the Tomcat server, and it is disabled by default. It works in the same way as described in Handling headers used for forwarding HTTP requests, but it can be used even if there is no proxy.

To specify a range of whitelisted IPs, use the `spring.cloud.gateway.mvc.trusted-proxies` property in the `<IA_ROOT>/config/iawebapp/application.yml` configuration file.

Examples of acceptable regular expressions can be found in Handling X-Forwarded-Host and host headers.

Important

The expression must match the IP or the hostname of the Gateway itself. If there are proxies in front of the gateway, the expression must also match their IPs.

Chapter 10

Upgrading OpenText Information Archive

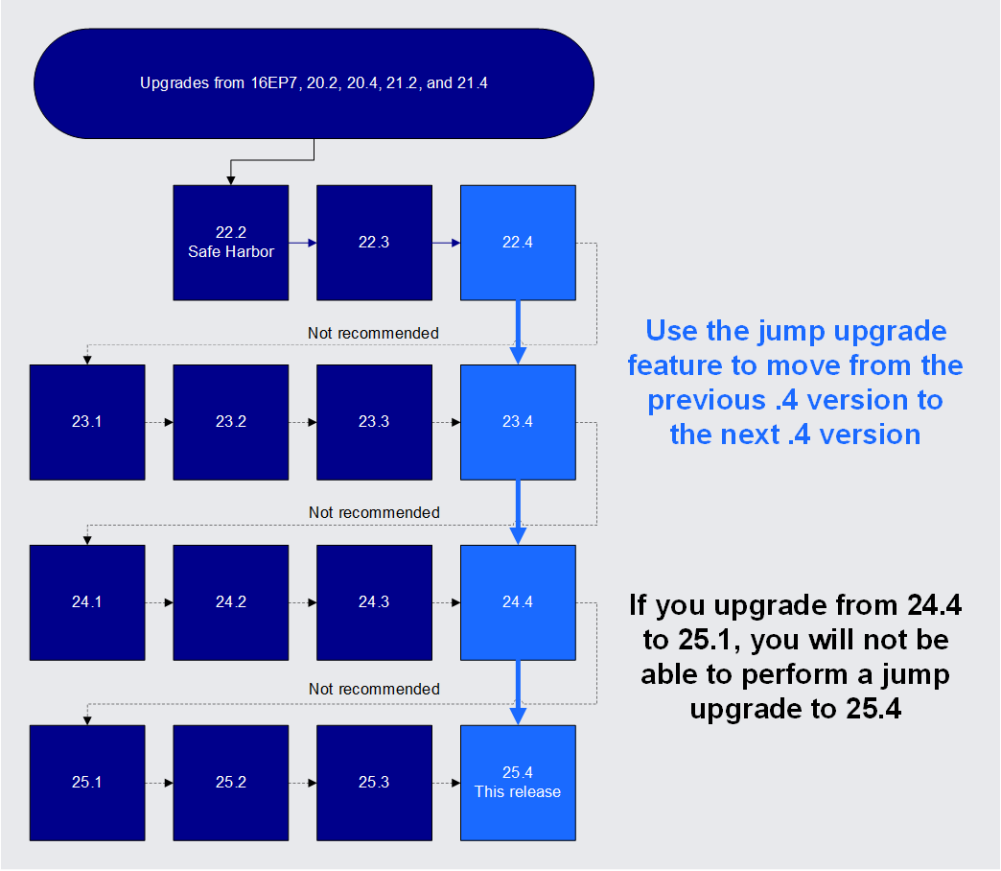
10.1 Supported upgrade paths

The following scenarios describe the required actions to upgrade an OpenText Information Archive deployment to the latest version. Notice that you are able to upgrade from the 23.4 version to the 24.4 version. If you upgraded from 24.4 to 25.1 and want to upgrade to 25.4, you must complete all the intermediate upgrades (25.2, 25.3) before upgrading to 25.4.

If you have a 22.4 system, you only require three upgrades, one from 22.4 to 23.4, another from 23.4 to 24.4, and another from 24.4 to 25.4.

Note: Perform a jump upgrade unless you have an important reason you need to upgrade to each consecutive version.

Be sure to migrate to the latest patch of the version being installed.



Your Existing Deployment Version	Upgrade Path
25.3	Follow the tasks described in this document.
25.1 through 25.2	Complete all the steps to upgrade to each incremental version until you reach 25.3, then follow the tasks described in this document.
24.4 *	Follow the tasks described in this document to perform a jump upgrade to the current version.
24.1 through 24.3	Complete all the steps to upgrade to each incremental version until you reach 24.4. Next, follow the tasks described in this document to perform a jump upgrade to the current version.
23.4 *	Follow the tasks described in the 24.4 *OpenText Information Archive Installation Guide* to perform a jump upgrade from 23.4 to 24.4. Next, follow the tasks described in this document to perform a jump upgrade to the current version.
23.1 through 23.3	Complete all the steps to upgrade to each incremental version until you reach 23.4. Follow the tasks described in the 24.4 *OpenText Information Archive Installation Guide* to perform a jump upgrade from 23.4 to 24.4. Next, follow the tasks described in this document to perform a jump upgrade to the current version.
22.4 *	Follow the tasks described in the 23.4 *OpenText Information Archive Patch Notes* to perform a jump upgrade from 22.4 to 23.4 directly. Next, follow the tasks described in the 24.4 *OpenText Information Archive Installation Guide* to perform a jump upgrade from 23.4 to 24.4. Next, follow the tasks described in this document to perform a jump upgrade to the current version.
22.2 through 22.3	Complete all the steps to upgrade to each incremental version until you reach 22.4, and then follow the step above.
16EP7 through 21.4	Follow the instructions in the *OpenText Information Archive Upgrade and Migration Guide*, which is specific for upgrading to the safe harbor (22.2) release only. Next, continue to complete all of the steps in the rows above. Jump upgrades from these versions only are supported to 22.2.
4.0 – 16 EP5.1	Complete all the steps to upgrade to each incremental version until you reach 16EP7. Then, follow the instructions for the row above.
Earlier than 4.0	You are strongly encouraged to engage OpenText Professional Services to help with your migration project.

*** Perform a jump upgrade unless you have an important reason you need to upgrade to each consecutive version.**

Note: The supported versions of Helm upgrade are different than the above. Refer to *OpenText Information Archive - Cloud Deployment Guide (EARCORE-ICD)* for more information.

10.2 Planning an upgrade

By planning your upgrade of OpenText Information Archive, you can help the upgrade process go more quickly and smoothly. The following questionnaire is meant to help you make these planning decisions.

10.2.1 Upgrade questionnaire

Before you upgrade OpenText Information Archive, consider the following questions about your existing installation:

- Are you able to upgrade all your OpenText Information Archive components in the same maintenance window? It is necessary to upgrade all components across all hosts before starting any up again. Make sure that you are considering all system components, which might be distributed over different computers and images, including dedicated IA Shell instances.
- Are you running IA Web App on external Apache Tomcat? When you upgrade OpenText Information Archive, it is recommended to briefly install IA Web App as a standalone Spring Boot application. This allows you to troubleshoot upgrade issues more easily. You can make sure that IA Web App is running properly and connected to all your other OpenText Information Archive components, and then deploy IA Web App to Apache Tomcat.
- Do you have enough disk space for the upgrade?
- Are you running multiple OpenText Information Archive components on the same computer with different settings for password encryption? OpenText Information Archive does not support generating the upgrade configuration simultaneously for multiple components on the same computer with different settings for password encryption. Instead, you must generate the upgrade configuration separately for each component.
- Are you planning on using SSL/TLS? This can be configured for database clusters or the IA Server/IA Web App.
- Are you using a shared keystore and truststore on a network location, or separate, local keystores and truststores for each component?
- Are you using OTDS to authenticate OpenText Information Archive users? If so, make sure to upgrade to the latest supported version of OTDS, including applying any required patches. For more information about certified environment and system requirements, see *OpenText Information Archive Release Notes* on support.opentext.com (https://support.opentext.com/).

- Are you using OTDS in SSO mode? If so, because the CLIENTS profile is not enabled, OpenText Information Archive ignores the contents of the `<IA_ROOT>/config/iawebapp/application-CLIENTS.yml` file. But you should still update the following parameters in this file so that when you upgrade OpenText Information Archive, the setup will use the correct values.:
  - Set the `clientSecret` for `infoarchive.jdbc` to the secret token for JDBC.
  - Set the `clientSecret` for `infoarchive.cli` to the secret token for IA Web App.

    These values were specified during the setup process.

    Important

    If you are using encrypted passwords, these should be the encrypted versions of these secrets.
- Are you are upgrading from a hot fix? If so, you should do the following to ensure fixed issues do not regress due to the upgrade to a non-latest patch version:
  - If there are patches available for the version of OpenText Information Archive that you are upgrading to, make sure that you are upgrading to the newest patch.
  - Make sure that all the bug fixes that were delivered in the hot fix are also included in the version of OpenText Information Archive that you are upgrading to. To verify this, see the patch notes

10.2.2 Upgrading from the latest version

The following instructions describe how to perform an upgrade from the previous OpenText Information Archive version, which can be on any patch of that version. You must perform some steps manually. Always follow industry standard best practices to back up your environment before making any changes

Important

- Before starting, ensure that there is no ingestion currently happening. Also ensure that no jobs are running and suspend the schedules for all jobs.
- The search templates for the Audit and Reports OpenText Information Archive applications may be updated. If you had made changes to the existing searches, you may want to either export or copy them. Otherwise, the upgrade replaces any modifications that you made with the default searches that come with any new OpenText Information Archive installation.
- Before you apply any changes to your production environment, you should thoroughly test all the changes in a development, testing, or staging environment.
- You should take a backup image of each of the servers that OpenText Information Archive runs on. In the event of a failure, server backup images

can help prevent data loss by giving you the ability to restore your system and roll back any changes that have been made to your servers, their local data, their configuration files, and their keystores.

The directory for the previous version of OpenText Information Archive is represented by the variable `<IA_UPGRADE_FROM_ROOT>`. The directory for the new version of OpenText Information Archive is represented by the variable `<IA_UPGRADE_TO_ROOT>`. `<IA_UPGRADE_TO_ROOT>` is expected to be separate directory from `<IA_UPGRADE_FROM_ROOT>`.

If you want to migrate to a new computer as part of the upgrade, then `<IA_UPGRADE_TO_ROOT>` should be a directory on the new computer.

- Relative path (for example, data/root). If you have multiple instances of IA Server, this needs to be a network location.

  Important

  If you change this path, make sure that it points to the same resolved location as the original path. If you change this path to a different location, OpenText Information Archive will not be able to find your data because changing this path does not move data from one location to another.

- If you installed the existing OpenText Information Archive components as system services, uninstall the system services. For more information, see Uninstalling OpenText Information Archive System Services.

10.2.3 Overview for upgrading a production configuration

In a production configuration, OpenText Information Archive components are installed on separate computers. There are six tasks for upgrading OpenText Information Archive. Immediately below are high-level descriptions of these tasks. For the step-by-step instructions, see Performing the upgrade tasks.

1. **Generate the OpenText Information Archive configuration**: On each computer that hosts a system component, generate the configuration for OpenText Information Archive components, which determines how the upgrade proceeds. This can be done while the existing system components are still running. Once the configuration has been generated on each computer, stop all existing components.

   If you are running with SSL, the setup program will warn of any keystores that are in the old installation. It is recommended that they be copied over to the new location to avoid accidently removing them during cleanup of the older version.

2. **Shutdown your existing system and take a backup**.

3. **Finalizing the upgrade configuration**: This section goes over some optional steps for starting the components as services or configuring the IA Web App to be installed on Tomcat.

4. **Start the new version of OpenText Information Archive**: Start the system components in a particular set order (first the database, then the server, then the web application).

5. **Upgrade the first-time applications:** On the IA Shell instance that you want to use to install OpenText Information Archive applications, upgrade the first-time setup applications. Also, complete the few remaining tasks necessary for the upgrade.

6. **Verify that the system upgrade is ok.**

Important
If you are using encrypted passwords in configuration files, or TLS/SSL, or have chosen to persist the crypto object keystore on a filesystem (which is strongly discouraged), then there are corresponding parameters in configuration files that specify the locations of keystores and truststores. When you upgrade, you must make sure that the keystores and truststores are still accessible at the paths specified in the configuration files for OpenText Information Archive.

Note: Although it would be technically possible to upgrade a demo configuration, such an upgrade will require additional manual steps and is not officially supported, nor recommended. This is therefore not covered in this section.

10.3 Performing the upgrade tasks

Follow all the steps in the following sections to perform the upgrade.

10.3.1 Generate the OpenText Information Archive configuration

Make sure that all environment and system requirements are met, as described in *OpenText Information Archive Release Notes* on support.opentext.com (https://support.opentext.com/). This includes installing the appropriate version of JDK and making sure that the `JAVA_HOME` environment variable is set to the directory for JDK (for example, `C:\jdk`).

On each computer that hosts an OpenText Information Archive component to be upgraded, complete the following:

1. Extract the OpenText Information Archive ZIP file to the `<IA_UPGRADE_TO_ROOT>` directory.

   - Run the following install script in the `<IA_UPGRADE_TO_ROOT>` directory, rather than the setup or `iasetup` scripts in the `<IA_UPGRADE_TO_ROOT>/bin` directory:

     - Windows: `install.bat`

- Linux: `./install`

The install script generates the upgrade configuration and applies the password encryption settings from one of the components to all the components.

2. If you are upgrading using the `install` command, the following prompts appear:

   a. `Install OpenText Information Archive using a demo configuration? (Y/n)`

   Type **N** and press **ENTER**.

   b. `Upgrade from an existing OpenText Information Archive installation? (Y/n)`

   Type **Y** and press **ENTER**.

3. The following prompt appears:

   `Shared resources location: (<IA_ROOT>/iadata)`

   Tip: If you are unsure of the location of the `iadata` folder, its location is listed in the following files in the `<IA_UPGRADE_FROM_ROOT>` directory:

   - The `transcript.txt` file lists the location (`Shared resources location`).
   - The `application-apply.yml` file includes a `sharedLocation` property that lists the folder's location.

4. The following prompt appears: `Location of existing OpenText Information Archive installation:`

   Specify `<IA_UPGRADE_FROM_ROOT>`.

   Note: The directory specified here must contain at minimum the `config` directory with subdirectories per applicable component to upgrade for the new configuration files to be generated properly.

5. The following prompt appears: `Save this setup configuration as a template: (Y/n)`

   The template saves your answers to the questions asked above to the `<IA_UPGRADE_TO_ROOT>/config/iasetup/application-apply-upgrade.yml` file. Optionally, you can then run the following command, which will go through the questions again with your previous answers as the defaults:

   - Windows: `<IA_UPGRADE_TO_ROOT>\bin\iasetup apply-upgrade --config.review= true`
   - Linux: `<IA_UPGRADE_TO_ROOT>/bin/iasetup apply-upgrade --config.review=true`

Note: It is strongly recommended to save your setup configuration as a template. The command above is the easiest way to make changes to the configuration of an existing upgrade.

Important

The template saves any passwords that you specify in clear text (unencrypted). Storing passwords and tokens in clear text is a potential security risk. After you finish using the `<IA_UPGRADE_TO_ROOT>/config/iasetup/application-apply-upgrade.yml` file in the upgrade process, you should consider removing the entire file or at least remove the unencrypted passwords and tokens from the file.

However, once you do so, the file can no longer be used to reapply and generate the upgrade configuration again.

6. The following prompt appears: `Initialize configuration (generate actual configuration files): (Y/n)`

   Type `y` to generate the configuration files. Without these files, you cannot run the system components. You can type `n` if you realize that you want to change the configuration, and you are going to generate the upgrade configuration later.

   The configuration files are generated, and the upgrade software finishes.

   Notes

   - Review and address any other warnings that you see.
   - If you run the install command again, and you previously chose to generate the configuration, then the software will replace the configuration that you generated before with a new one.

Important

The setup program generates configuration based on your previous version of OpenText Information Archive. Be sure to review the configuration generated by the setup program and compare with your previous configuration. If you have made any changes to a configuration file directly after running setup, check to ensure the configuration was generated correctly.

If you are using OTDS, upgrade to the latest supported version of OTDS, including applying any required patches. For more information about certified environment and system requirements, see *OpenText Information Archive Release Notes* on support.opentext.com (https://support.opentext.com/). Follow these instructions to ensure that the OpenText Information Archive partition is created.

Important

If you made any manual changes to configuration files in the earlier version, any updated properties will not be included in the generated configuration. Always check any overrides that you may have done to see if they were included Any custom Zuul configuration that was added is ignored as of 23.4 P01 and above, as the product no longer uses the Zuul libraries.

If you updated the 25.2 or 25.3 version of the `spring.gateway.mvc` section in the `<IA_ROOT>/config/iawebapp/application.yml` file, you must manually update the section again for the 25.4 version. During development of the 25.4 version of the product, the section was renamed to `spring.gateway.server.webmvc`. Due to technical limitations, this particular section does not inherit any manual changes made in 25.2 or 25.3 during an upgrade to 25.4, and the changes need to be redone manually instead.

10.3.2 Shut down your existing system

After you have generated the configurations for all the OpenText Information Archive components in your existing system, do the following in this order:

1. Close all IA Shell sessions.
2. Stop all instances of IA Web App.
3. Stop all instances of IA Server.
4. Stop all instances of PostgreSQL Server.

10.3.3 Uninstalling OpenText Information Archive system services

If the components of your installation have been distributed over multiple servers, hosts, or nodes, you must remove each one of those services separately.

Note: OpenText Information Archive does not configure PostgreSQL to run as a service.

To uninstall system services for OpenText Information Archive components:

- Open a command prompt in the `<IA_ROOT>/bin` directory, and then run the following command:
  - Windows: `setup.bat --no-config --remove`
  - Linux: `./setup --no-config --remove`

10.3.4 Upgrading on-premises PostgreSQL instances

If you need to upgrade on-premises PostgreSQL instances, refer to the documentation on the PostgresSQL website. If the data files of the new instance are on the same storage as the old instance, use the `--link` option to avoid data duplication.

Note: The upgrade procedure does not update the runtime properties stored in `postgresql.conf` file. It should be done manually after the upgrade is complete.

Ensure that on each IA Server, the `IAPSQL_HOME` environment variable points to an also upgraded local PostgreSQL (client) installation.

If there are multiple instances used for different types of data (system data and structured data), all instances should be upgraded to the same version before IA Server is started again. If the new instances' ports were changed, the IA Server configuration should be updated. The system data PostgreSQL settings (`systemData: psql: databaseCluster: url`) should be updated in the `config/iaserver/application.yml` file before the IA Server is started up and, for the structured data, each individual node should be updated on the **Administration** page after the startup.

10.3.5 Ensuring the application-otds.yml file is configured correctly

If you had modified the `<UPGRADE_ROOT>/config/iaserver/application-otds.yml` file, it is recommended that you ensure the values are correct, as setup generates this file based on your previous version.

Some changes may need to be done to the `application-otds.yml` file.

If you wish to use `https` to connect to OTDS, change the `protocol` value to `https` and set the `port` value to what you previously had as the `httpsPort` value.

Further changes are required to enable Content Aviator. For more information, see section 3.12.1 "Enabling Content Aviator" in *OpenText Information Archive - Administration Guide (EARCORE-AGD)*. Note that you will need to create an OAuth2 client in OTDS with the name `infoarchive.server` and specify the `clientSecret` in this configuration file. The `authUrl` value is the URL that IA Server will use to generate a token to communicate with Aviator.

In the following example, the OAuth2 client that was added is `infoarchive.server`. The `clientSecret` value is case sensitive.

If you are not planning on using Aviator, leave the `clientSecret` value empty for an upgrade.

10.3.6 Finalizing the upgrade configuration

1. If the IA Server instances for your existing OpenText Information Archive deployment do not map a network filesystem to the `<IA_UPGRADE_FROM_ROOT >/lib/iaserver/external` directory, then perform the following steps.

   Note: If you want to avoid performing the following manual steps during future upgrades, you should map a network filesystem to the external directory.

   a. Check whether the IA Server instances in your previous OpenText Information Archive deployment have any files in their `<IA_UPGRADE_FROM_ROOT>/lib/iaserver/external` directories.
   b. Copy these files to the `<IA_UPGRADE_TO_ROOT>/lib/iaserver/external` directory for each IA Server instance for the new OpenText Information Archive.

2. If language support or branding was customized for your previous version of OpenText Information Archive, but the customization location for IA Web App was not set to a location on a network filesystem, then copy the contents of the `<IA UPGRADE_FROM_ROOT >/config/iawebapp/` customization directory into the `<IA_UPGRADE_TO_ROOT>/config/iawebapp/` customization directory on each of the computers that hosts IA Web App.

3. Optional Deploy IA Web App to Apache Tomcat. For more information, refer to Deploying IA Web App to Apache Tomcat.

10.3.7 Starting the upgraded version

To start the upgraded version of OpenText Information Archive:

1. If you are not using an external PostgreSQL Server, start all the OpenText Information Archive instances of PostgreSQL Server by opening a command prompt, going to the `<IA_UPGRADE_TO_ROOT>\psql` directory, and running the following command:

   - Windows: `bin\pg_ctl.exe -Dconf start`
   - Linux: `bin/pg_ctl -Dconf start`

   At this point, you are starting these components from command prompts so that you can check for any errors.

2. Start one of the OpenText Information Archive instances of IA Server by opening a command prompt, going to the `<IA_UPGRADE_TO_ROOT>` directory, and running the following command:

   - Windows: `bin\iaserver`
   - Linux: `bin/iaserver`

   This instance of IA Server then performs synchronous upgrade processing, which blocks the startup of other instances of IA Server until the processing is done. You can track the progress of the synchronous upgrade processing in the command prompt.

   Notes

   - At this point, you are starting this component from a command prompt so that you can check for any errors.
   - If you have multiple instances of IA Server, you should let the first one come up before you start the other instances. If you do not, you will get a message that the upgrade has not finished. You do not have to do anything to address this error message.
   - You will see a warning if you have not installed the first-time setup applications. If you have not installed them yet, do so.

3. Start all the OpenText Information Archive instances of IA Web App by opening a command prompt, going to the `<IA_UPGRADE_TO_ROOT>` directory, and then running the following command:

   - Windows: `bin\iawebapp`
   - Linux: `bin/iawebapp`

   At this point, you are starting these components from command prompts so that you can check for any errors.

Important

It is important to view the status of the Upgrade jobs because, if they do not succeed, upgrade to the next version will be blocked. It is strongly recommended that you track that the Upgrade jobs have passed.

10.3.8 Upgrading the first-time applications

Upgrading the first-time applications ensures the value lists for the new audit event types are created.

Only upgrade the first-time applications if you see that the Audit and Reports applications have been installed.

To upgrade first-time applications:

Open a command prompt, go to the `<IA_UPGRADE_TO_ROOT>/first-time-setup/applications` directory, and run the following command:

- Windows: `upgrade.bat`
- Linux: `./upgrade`

10.3.9 Adding OpenText Information Archive applications from previous versions of OpenText Information Archive

You are only allowed to install applications created for OpenText Information Archive 22.2 or later. You cannot install an application that used the legacy database architecture.

10.3.10 Reregistering system services

As the paths have changed, you need to reregister the system services, if required. For more information, refer to Installing system services.

10.4 Verifying the upgrade

To verify that the system upgrade is working:

1. Clear your browser cache and force a refresh.
2. Login to IA Web App and verify that everything is working properly by doing the following:

   a. To verify the version number of OpenText Information Archive, click  and select **About OpenText Information Archive**.

      Ensure there is a value for the **IAWA Version**.

   b. Test searches for each application, including the event name and type for audit searches.

      Navigate to the **Administration > Jobs** page to ensure the UPGRADE: Execute Asynchronous Upgrade Tasks and UPGRADE: Execute Synchronous Upgrade Tasks jobs ran without errors. You should see the word **Success** in the **Status** column. Click the name of a job to view a detailed status and click the **Success** link to view the logs and look for any warnings.

There is a system-level audit event for upgrade. This audit allows you to verify when an upgrade was performed, and to which version. The upgrade audit event is always enabled.

The following procedure should be completed only if you are using OTDS:

1. Start the Tomcat instance hosting OTDS and let it finish.
2. To verify this was done correctly, login using the OTDS Admin client and click **Partitions**. There should be one `inforarchive.bootstrap` partition.

To see the audit when OpenText Information Archive was upgraded:

1. Run the Archive Audits job.
2. In IA Web App, click **Applications > Audit > System Audit**.
3. In the **Event Type** list, choose **System**.
4. In the **Event Name** list, choose **Upgrade**.
5. Click **Search**.

   To see which version the upgrade is for, expand the search result by clicking the arrow icon to the left of the search result.

To check the OpenText Information Archive version:

- In the IA Web App, select **About**.

Tip: If the version is incorrect, complete a hard reset on your browser to clear the browser cache and retry the About command.

Note: The version of PostgreSQL is for the system data node (cluster).

Chapter 11

Upgrade troubleshooting

The command prompt window displays installation and upgrade status messages, including errors that may occur during the installation or upgrade. You can also find status messages in the log files in the `<IA_ROOT>/logs/iasetup` directory. You should look at the log files even after a seemingly successful installation or upgrade.

Note: You can locate errors by searching for occurrences of the word `WARN` or `ERROR` in the log files.

For more information, see section 3.13 “Logging” in *OpenText Information Archive - Administration Guide (EARCORE-AGD)* and its subsections.

11.1 Cannot start IA Server after upgrade

Issue: **IA Server fails to start after upgrade.**

1. Inspect and analyze the error message(s), as these may provide sufficient context and detail to identify the problem.
2. Ensure that IA Server is not already running and check the requested port.
3. Ensure that the `system PostgreSQL` database Server is running.
4. If using PostgreSQL SSL, ensure the SSL sections in the `postgresql.conf` and associated sections in the IA Server's `application.yml` files are configured correctly.
5. Check the settings in the `pg_hba.conf` file, and ensure the method is set properly (the following is merely an example and will look different in production:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
```

   The method may have different values and, depending on whether you want no `SSL`, `1–way SSL`, or `2–way SSL`, the values have to match.

   Ensure that PostgreSQL is configured to enable remote access. It is not recommended to try to connect using localhost. Use the IP address when connecting to PostgreSQL.
6. If passwords are encrypted, ensure the appropriate configuration files are in the expected locations (`truststores` and `keystores`).

- Check that the `<IA_ROOT>/config/iaserver/application.yml` file points to the correct location of keystores and data.

- Look at the upgrade template and transcript and verify that absolute paths are used in the configuration files for the `iadata` directory, instead of relative paths.

Issue: **IA Server cannot connect to system databases.**
Ensure the database cluster configuration is correct and check the status of the database. If using SSL, ensure that the files are in the expected location.

Issue: **IA Server cannot access unstructured content.**
Check that the data root for the file system is not a relative path and that the configured path is accessible to the IA Server.

To change the location of the fileSystemRoot to an absolute path, do the following:

1. In IA Web App, go to **Administration > Storage**.
2. In the **Storage Systems** section, next to **defaultFileSystemRoot**, click  and select **Edit**.

   Important
   If you change this path, make sure that it points to the same location as the original path. If you change this path to a different location, OpenText Information Archive will not be able to find your data because changing this path does not move data from one location to another.

You can also follow these steps if having difficulty starting IA Server after an upgrade:

1. If using SSL for the server, ensure the required keystores and truststores can be found with the generated `application.yml` files. Setup will not copy these files from the previous installation and explicitly warns about this condition, if noticed. Check the transcript and the setup logs.
2. When upgrading, if PostgreSQL database has not been initialized (meaning the server did not start up), starting up the server results in an error that the keystore could not be found. Either have the PostgreSQL system database correctly installed or do a clean install (you will not be able to upgrade an empty system database).
3. If you are upgrading a demo configuration (not recommended) or single node deployment, ensure that the data for structured data is available.

11.2 PostgreSQL Server does not start

Ensure that there are no relative paths in the `data_directory` for the `postgresql.conf` file in the `<IA_UPGRADE_TO_ROOT>/psql` directory and that the path is correct.

11.3 Issues with IA Shell after an upgrade

Issue: **IA Shell is not working after the completion of the upgrade procedure.**
It is strongly recommended that, when changing the IA Shell's `application.yml` file, that the `application-CLIENTS.yml` file in the IA Web App directory is also changed. On the next upgrade, the generated configuration will use the value from the `application-CLIENTS.yml` and the IA Shell `connect` command will fail.

Issue: **Some IA Shell commands are no longer available.**
Some commands that were specific to the legacy architecture are no longer available.

11.4 Running the OpenText Information Archive components after upgrade

Issue: **OpenText Information Archive components fail to start after upgrade.**

- If some OpenText Information Archive components that are installed on the same computer start, but some do not start, you might have configured different encryption settings for the components. You can correct this by making sure that if one component indicates the passwords are encrypted, all other components use the same settings and also use encrypted passwords.
- Make sure that all the YAML files in the `<IA_ROOT>/config` directory structure are well formed. When you enable or disable a profile in an `application.yml` file, the profile must be indented by two spaces relative to include. In other words, the dash (`-`) character in the profile that you enable must line up with the `c` character in the include line that occurs earlier in the file.

11.5 Starting IA Web App after upgrade

Issue: **IA Web App fails to start after upgrade.**
Look in the `<IA_ROOT>/config/iawebapp/application.yml` file to see if any relative paths are specified. Change the relative paths to absolute paths.

Issue: **Cannot login to IA Web App when using OTDS SSO.**
If you cannot login, verify the configuration settings defined in the `config/iawebapp/application-infoarchive.gateway.profile.OTDS.yml` file are correct. Specifically, ensure the `OTDS.password` is set.

If you are using external Tomcat, ensure that the `infoarchive.gateway.contextPath` and `infoarchive.gateway.port` are set correctly in the `config/iawebapp/application.yml` file, and the values match redirect URL, which can be viewed and updated in OTDS Admin for the `iawa` OAuth Clients.

11.6 Running first-time setup scripts or installing OpenText Information Archive applications after upgrade

Issue: **Cannot run scripts for first-time setup applications or other OpenText Information Archive applications. Cannot log into IA Shell when running first-time setup applications, and cannot log into IA Web App.**

- Make sure that all required instances of the database cluster, IA Server, and IA Web App are running.
- If you cannot log into IA Shell while using the script, check the credentials of the user and make sure that the correct profiles are enabled in the `<IA_ROOT>/config/iawebapp/application.yml` file.
- Using a text editor, open the default.properties file for the first-time setup applications (in the `<IA_ROOT>/first-time-setup/applications` directory) or the other OpenText Information Archive applications (in the `<IA_ROOT>/examples/applications` directory).

  Then do the following:

  – Check that the `rdbDataNode.bootstrap` parameter is correct.
  – Check that the value for the `rdbDatabase.adminPassword` parameter complies with the password strength requirements for `RdbDatabase` objects.
  – By default, this is the default password for every OpenText Information Archive `application.properties` file for the application (for example, `<IA_ROOT>\examples\ applications\PhoneCalls\config\configuration.properties`).
- Check that the OpenText Information Archive Gateway secret keys match for IA Server and IA Web App:

File	Parameter
<IA_ROOT>/config/iaserver/application.yml	infoarchive.gateway.token.secret
<IA_ROOT>/config/iawebapp/application.yml	infoarchive.gateway.client.token.secret

- If you are not using OTDS in SSO mode, check that the OpenText Information Archive Gateway client secrets match for IA Shell's main configuration file and the IA Web App clients file:

File	Parameter
<IA_ROOT>/config/iashell/application.yml	gateway.clientSecret
<IA_ROOT>/config/iawebapp/application-CLIENTS.yml	clientSecret in the clientId: infoarchive.cli section

  If you are using OTDS in SSO mode, these parameters should be different because OTDS acts as the OAuth2 authorization server and issues the JWT token. For more information, refer to configuring OTDS for SSO mode.

- If passwords were encrypted, make sure that all of the values in the configuration files were updated correctly.

Issue: **You can no longer install applications.**

To resolve the issue:

1. Update the scripts to install DC applications, as the `--cmdfile` option no longer works.

   Before:

   ```
   @echo off
   call ..\..\..\bin\iashell --cmdfile install.iashell --cmdfile_echo
   ```

   After:

   ```
   @echo off
   call ..\..\..\bin\iashell script install.iashell
   ```

2. Remove the last line, which is the `quit` command from the `install.iashell` script, because it is not supported in non-interactive mode.

You receive an error when trying to install applications after an upgrade.

If you receive an exception like the following, the easiest solution is that you reference the RDB datanode rather than specifying the definition. You should not follow the suggestion to use `create_or_update`, as this can result in unintentional configuration changes:

```
java.lang.IllegalStateException: Failed to execute ApplicationRunner
Caused by: com.emc.ia.cli.exception.CliScriptRunnerException: Could not run command
```

```
'import Reports/config'
Caused by: org.springframework.web.client.HttpClientErrorException: 400 errors:
Error: /rdbDataNodes/structuredData
Message: Object with name 'structuredData' of type 'RdbDataNode' already exists,
but with different values for attributes: [connectionProperties={sslkey=, sslcert=,
sslpassword=, sslrootcert=}:{sslmode=, sslpassword=, sslkey=, sslcert=, ssl=false,
sslrootcert=}]. Use 'configure: create_or_update' in order to update object.
 on POST request for "http://localhost:8765/systemdata/configuration"
```

You may also receive similar errors for `fileSystemRoot` if it had been changed and the `iashell/config/default.properties` file was not updated.

11.7 Viewing packages after upgrade

Issue: **After an upgrade, you cannot view packages for a SIP application.**

In IA Web App, go to **Applications > [Application Name] > Packages**. Look for an error like the following:

```
Internal Server Error.

javax.naming.ConfigurationException: E:\infoarchive\data\root\ PhoneCalls-space-
root-folder_fd7bce64-580b-42f7-bd5b-88eb22d42e63\ stores\default-store is not a
directory : E:\infoarchive\data\root\
PhoneCalls-space-root-folder_fd7bce64-580b-42f7-bd5b-88eb22d42e63\stores\default-
store
```

To fix an error like this, make sure that you are using a defaultFileSystemRoot with an absolute path by doing the following:

1. In IA Web App, go to **Administration > Storage**.
2. In the **Storage Systems** section, next to `defaultFileSystemRoot`, click the  actions icon and choose **Edit**.
3. Make sure that the Folder Path is an absolute path (for example, `\\server01\data\root` or `E:\iadata\root`) rather than a relative path (for example, data/root). If you have multiple instances of IA Server, this should be a network location

   Important

   If you change this path, make sure that it points to the same location as the original path. If you change this path to a different location, OpenText Information Archive will not be able to find your data because changing this path does not move data from one location to another.

11.8 Unexpected behavior after upgrade

Issue: **After an upgrade, IA Web App behaves unexpectedly (for example, text is missing from the screen).**

Do the following:

1. In your browser, force a refresh of IA Web App or open a private window and log into IA Web App.
2. In IA Web App, in the top-right corner of the page, click  and select About OpenText Information Archive.
3. Confirm that the version numbers are as expected

Issue: **After an upgrade, you encounter issues updating the group/role mappings when using OTDS.**

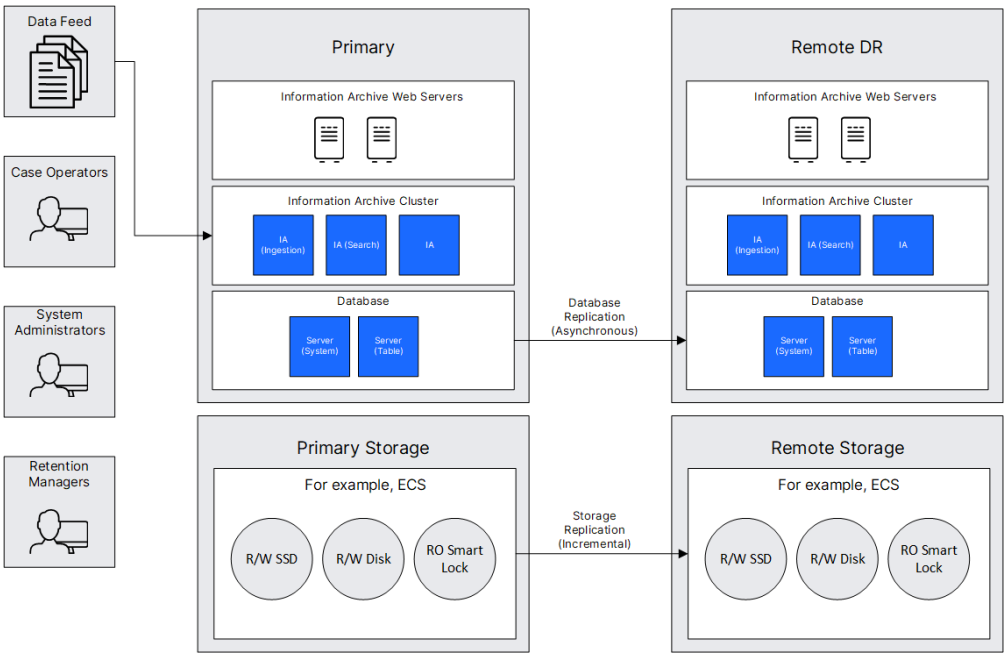
Do the following:

1. Check the Ensuring the application-otds.yml file is configured correctly section.
2. If any changes were necessary, restart the IA Server for the changes to take effect.

Chapter 12

Disaster recovery

The following is the supported disaster recovery setup for OpenText Information Archive. There is a primary and remote configuration, PostgreSQL is replicated asynchronously, and storage replication is incremental.



12.1 Requirements

The following is required for successful failover to the disaster recovery site:

- Asynchronous PostgreSQL data-node (server) replication for:
  - System data (including audit data)
  - Optionally, structured table archiving data (metadata)
  - You can map system and audit data to different PostgreSQL data nodes. For simplicity, we recommend using the same data for your system and audit data. You must nevertheless store your data in different databases within that data node.
- Storage replication of unstructured data. This includes backup stores:
  - Holding:
    - `libraryBackupStore`

    - `retentionBackupStore`
  - Database:
    - `retentionBackupStore`
- The following items must be seamlessly accessible on the disaster recovery site:
  - Mount points (the paths of `FileSystemRoots`)
  - CAS endpoints (connection strings of `ContentAddressedStorage` objects)
  - Custom Storage endpoints (property bags of `CustomStorage` objects)
  - ECS/S3/ For example, endpoints and credentials (`StorageEndPoint` and `StorageEndPointCredential` objects)

Caution

File system storage replication must preserve file system paths.

12.2 Other options

Here are some other recovery options to consider:

- Regular backups of the system data, including the System and Audit data. A repository must be made for this data. How you approach this option is dependent upon your setup.
- Regular backups of the retention data. This option only applies if no Retention Backup Store is configured for your holdings and databases. We strongly recommend that you do configure such stores though.

12.3 Disaster recovery and failover steps

As mentioned before, a requirement for failover is that you replicate the PostgreSQL node for system data and, optionally, selected nodes for a structured table data node. Refer to the PostgreSQL documentation to learn how to do this.

In 25.4, the following recovery process was introduced:

Prior to completing the following procedure, shut down everything on Primary (if the disaster did not already do that).

Complete the following steps to perform failover on the disaster recovery site:

1. Turn the System Data and any structured data PostgreSQL replicas into masters.
2. Start the new PostgreSQL servers of these masters on the desired hosts and ports:
   - System data (and audit data) PostgreSQL server

- Any structured data servers

If necessary, create new replicas of these servers.

3. Start the IA Server with the additional (YAML) configuration:

```
profiles:
  include:
    - infoarchive.ias.profile.no.cleanup.rollForward
    - infoarchive.ias.profile.recovery
recovery:
  name: disaster on nov 12
  days: 1
  hours: 12
```

Name your recovery configuration (for example, based on the date the disaster occurred) and add an indication of how far in the past it needs to look for potential inconsistencies to fix. For example, if replication of your storage systems is guaranteed to be less than 24 hours, 1 day is all you need to configure.

This run of the IA Server only sets up recovery and terminates immediately after. Check the console logging for messages on recovery setup.

4. Bring up your full regular deployment without the profiles above and without the recovery settings.

Because of the previous step, the IA Server starts executing the Recovery job in the background. During its execution, the IA Server works as normal, except for when it needs access to data partitions that still have potential things to fix that were not yet covered by the job. Those operations will fail until the underlying data partitions are “released” by the job.

5. After the job finishes, check its logs for what it recovered and for any warnings/errors.

In case of warnings/errors, the IA Server may not have been able to properly process all data partitions it planned to recover. These should be checked separately by, for example, a Business Owner or Retention Manager. Once done, run the Recovery job again with the setting **Leave Recovery Mode** to force the job to release any data partitions it locked

The former disaster recovery process still works:

1. Turn the System Data and any structured data PostgreSQL replicas into masters.
2. Start the new PostgreSQL servers of these masters on the desired hosts and ports:

- System data (and audit data) PostgreSQL server
- Any structured data servers

If necessary, create new replicas of these servers.

3. Modify the `YML` file to include the proper PostgreSQL Server IP addresses along with any other settings regarding the environment:

- System data (and audit data) PostgreSQL server

4. Start an IA Server that cannot be accessed publicly.
5. Start IA Shell from the `<IA_ROOT>/bin` directory, and connect it to IA Server.
6. Run the following command for any structured data that is configured in your system, and for which the bootstrap address on the DR site is different from the bootstrap address on the primary:

```
iashell> update /rdb-data-nodes/<DATA_NODE_NAME> --properties "'bootstrap' :
'<DR_BOOTSTRAP_ADDRESS>'"
```

7. Run:

```
iashell> recover-system
```

   The report of this recovery is only available in the server logs.

   AIPs that have become inconsistent are invalidated. In the worst case scenario, this may imply that a previously committed AIP was demoted.

   After this, the various components, repositories, or both of OpenText Information Archive should be consistent again.

   How long the command runs depends on the number of AIPs and tables with retention applied in your system. It should take several hours for 100,000 AIPs.

8. Start all necessary servers to resume production.

If the IA Server was in the middle of an application-level SIP disposition (meaning that an entire SIP application was under retention, and the IA Server was about to dispose it) when a disaster happened, then the IA Server may not properly recover. This means that (parts of) the application may reappear after the IA Server comes up on the disaster recovery site. In that case, you need to generate the purge list and run disposition again.

Disaster Recovery is more extensively covered in the *OpenText Information Archive Disaster Recovery* whitepaper which is part of the Champion's Toolkit on support.opentext.com (https://support.opentext.com/).

12.4 Recovering archived audits

Archived audits are not removed from the audit database until the time indicated for the `rollForwardData.rollForwardObjectRetention` setting has elapsed and the Archive Audits job is run again.

If a disaster occurs before the time indicated by the `rollForwardData.rollForwardObjectRetention` setting has elapsed, and it is not possible to automatically recover the archived audits because the backup of the packages containing them has been corrupted, the `recover-system` IA Shell command automatically invalidates those packages. Rerun the Archive Audits job to re-archive those audits.

12.5 Disaster recovery logging

Should disaster recovery be required, vital information about the recovery process can be found in the `recovery.log` file.

The `recovery.log` will indicate if it is not safe to re-enable background processing and start the other server.

A warning will indicate:

- Which applications are missing stores and
- Which stores are missing.

The warning also states that the recovery process did not act on anything for the specified applications.

The `recovery.log` contains important `INFO` messages to define which stage the system was in the processing (for example, recover holding, roll forward playback, post processing, etc.).

Any AIPs that could not be recovered are logged at WARN.

The `recovery.log` indicates how much time was spent on various tasks, as illustrated in the example in this section.

The following is an example of the `recovery.log`:

```
2022-11-30 13:48:06.135 [22] INFO   - Request for system recovery
2022-11-30 13:48:06.147 [22] INFO   - Step 1 of 3: Recovering holdings
2022-11-30 13:48:06.162 [22] INFO   - Starting recovery of holding 1b603b4a-
dbe0-48de-9b20-916fecdf5ec3 named 'PhoneCallsGranular'
2022-11-30 13:48:06.255 [22] INFO   - Completed processing of recovery of holding
1b603b4a-dbe0-48de-9b20-916fecdf5ec3 named 'PhoneCallsGranular'
2022-11-30 13:48:06.432 [22] INFO   - Starting recovery of holding
ba7b6a3d-65dd-4df2-9f47-9722693b1be7 named 'Audit'
2022-11-30 13:48:06.450 [22] INFO   - Data partition library with id '4d12d64a-72b3-407c-
b3e8-58e0a741d038' is not synchronized. Recovery starts.
2022-11-30 13:48:06.504 [22] INFO   - Data partition library with id 'c6cf4d3b-fe8f-40d3-
a376-3ec19f416cd9' is not synchronized. Recovery starts.
...
2022-11-30 13:48:07.295 [22] WARN   - Aip with id 'ca0db2b0-5514-452b-9b72-d95a9fd664b7'
failed to be recovered due to: Error restoring library
'4d12d64a-72b3-407c-b3e8-58e0a741d038' caused by: java.nio.file.NoSuchFileException: C:
\infoarchive\data\root\Audit-space-root-folder_ca83531c-b5cf-4bc8-
b7cc-7b1eadeba0a6\stores\default-store\4d\12\d6\4a-72b3-407c-
b3e8-58e0a741d038.library_tar_gzip.f86e7af0-3b9c-45f1-808d-79fb61f7e027
...
2022-11-30 13:48:07.299 [22] INFO    - Completed processing of recovery of holding
ba7b6a3d-65dd-4df2-9f47-9722693b1be7 named 'Audit'
2022-11-30 13:48:07.475 [22] INFO    - Step 2 of 3: Processing rollforward operations
2022-11-30 13:48:07.583 [22] WARN    - Inconsistencies found during check of aip.
Demoting to INVALID Aip external id =
e512eac8-2232-474a-8761-25fab6f46aa7 in holding: Audit
...
2022-11-30 13:48:08.582 [22] INFO    - Processed: 8 recent AIP commits, 8 records
rollfoward operations, 0 disposition operations
2022-11-30 13:48:08.582 [22] INFO    - Step 3 of 3: Post processing
2022-11-30 13:48:08.583 [22] INFO    - Replaying backup for applications requiring backup
2022-11-30 13:48:08.600 [22] INFO    - Recovery has finished processing. StopWatch
'System recover': running time = 2443637158 ns; [Recovering holdings]
```

```
took 1300272280 ns = 53%; [Find rollback operations] took 27817794 ns = 1%;
[Predispose] took 2199 ns = 0%; [Replaying rollforward operations] took 1107370254 ns =
45%; [Postdispose] took 3299 ns = 0%; [Replay backup] took 8171332 ns = 0%
```

Chapter 13

Appendix

13.1 Manual configuration

13.1.1 Enabling and disabling example user accounts

To enable example user accounts:

1. In a text editor, open the `<IA_ROOT>/config/iawebapp/application.yml` file.
2. Remove the # symbol from the following line:

```
# - infoarchive.gateway.profile.AUTHENTICATION_IN_MEMORY
```

Note: The dash (-) character in the above line must line up with the dash character in `- CLIENTS`.

To configure support for the LDAPs protocol:

1. Set up TLS/SSL for OpenText Information Archive components. For more information, see High-level overview: setting Up TLS/SSL for OpenText Information Archive components.
2. Import the certificate for the LDAP server into the truststore for IA Web App.
3. Add the following line:

```
- infoarchive.gateway.profile.AUTHENTICATION_EXTERNAL_LDAP
```

Note: The dash (-) character in the above line must line up with the dash character in `- CLIENTS`.

4. Configure external LDAP access.

   The following code sample demonstrates an Apache DS server running locally as a service in LDAPs mode. If you have your own LDAP set up, it must be configured in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.AUTHENTICATION_EXTERNAL_LDAP.properties` file:

```
# External LDAP
# The following two are default administrator and password for Apache DS
#AUTHENTICATION_EXTERNAL_LDAP.managerDn=uid=admin,ou=system
#AUTHENTICATION_EXTERNAL_LDAP.managerPassword=XXXXXX
# You can use | char as a separator and specify multiple userDNPatterns
AUTHENTICATION_EXTERNAL_LDAP.userDnPatterns=uid=\{0\},ou\=people
AUTHENTICATION_EXTERNAL_LDAP.userSearchFilter=uid=\{0\}
AUTHENTICATION_EXTERNAL_LDAP.userSearchBase=ou\=people
AUTHENTICATION_EXTERNAL_LDAP.groupSearchFilter=(member\=\{0\})
AUTHENTICATION_EXTERNAL_LDAP.groupSearchBase=ou\=groups
AUTHENTICATION_EXTERNAL_LDAP.url=ldap://localhost:10389/dc\=infoarchive,dc
\=opentext,dc\=com
```

5. Restart IA Web App.
6. Login to IA Web App with the `adam@iacustomer.com` example user account. The default password is `password`.
7. In IA Web App, navigate to **Administration > Groups** and verify that the groups from the external LDAP are shown there in the **Group Name** column.
8. You must assign the Administrator role to at least one LDAP group. You can also assign roles to other LDAP groups as desired.

To disable example user accounts:

1. In a text editor, open the `<IA_ROOT>/config/iawebapp/application.yml` file.
2. Comment out the following line by adding the `#` symbol to the beginning of the line:

```
# - infoarchive.gateway.profile.AUTHENTICATION_IN_MEMORY
```

3. Restart IA Web App.

13.2 Demo configuration

13.2.1 Working with example user accounts

In OpenText Information Archive, you can enable example user accounts, which are also known as in-memory user accounts. You would typically use these accounts with a demo configuration to test its features, set up a proof of concept, give a short demonstration, or set up a basic development environment. Each example user account has specific access rights that allow you to see how the system is accessed by a user with those access rights.

Caution

Do not use the example user accounts in a production environment. Attackers can use one of the example user accounts to gain unauthorized access to your OpenText Information Archive system and the assets that it contains. These out-of-the-box accounts are meant for demo and limited configuration purposes only, and the default password for each account is `password`.

The following table lists the account details for the example user accounts.

Username	Default Password	Role	Name
`adam@iacustomer.com`	`password`	Administrator	Adam
`bob@iacustomer.com`	`password`	Business Owner	Bob

Username	Default Password	Role	Name
connie@iacustomer.com	password	Developer	Connie
emma@iacustomer.com	password	End User	Emma
imran@iacustomer.com	password	IT Owner	Imran
rita@iacustomer.com	password	Retention Manager	Rita
audrey@iacustomer.com	password	Auditor	Audrey
sue@iacustomer.com	password	Administrator Business Owner Developer End User IT Owner Retention Manager	Sue

If you want to change the password for an example user account, you must update the password in the following file:

`<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.AUTHENTICATION_IN_MEMORY.properties`.

13.2.2 Setting up TLS/SSL in a demo configuration

To better understand how TLS/SSL works in OpenText Information Archive, you can set up TLS/SSL in a demo configuration. By using a self-signed certificate for IA Server, IA Web App, PostgreSQL server, and IA Shell, you can get TLS/SSL running quickly. Also, you can use a single certificate, keystore, and truststore because all of the components are on the same computer.

Caution

You should not use a self-signed certificate in a production configuration.

The following assumptions are made in the instructions below:

- You installed a demo configuration.
- You want to use the `<IA_ROOT>/config/https` directory to store the keystore and truststore. However, you can use any location that you like for the keystore and truststore as long as you specify the location correctly in the appropriate configuration files.

- IA Server, IA Web App, PostgreSQL server, and IA Shell are all running on the same computer.
- You want your keystore and truststore to be in JKS format. You can use PKCS instead, if you want. The instructions for setting up TLS/SSL in a production configuration use PKCS, so see those instructions for the necessary settings.
- You are using a Windows computer and Java Keytool to manage your keys and certificates. However, you can use a Linux computer and other software to manage your keys and certificates.
- You added the directory that contains the Keytool executable to your PATH system variable.

To prepare the self-signed certificate, keystore, and truststore:

1. Stop the OpenText Information Archive components.
2. Go to the `<IA_ROOT>/config` directory and create the `https` directory.
3. In a command prompt, go to the `https` directory that you just created, and then do the following:

   a. Create a self-signed certificate and a keystore for IA Server and IA Web App by running the following command (note that hostname should be the hostname for target machine (you will need one for each):

   ```
   keytool -genkey -noprompt -trustcacerts -keyalg RSA -alias
   ia_demo_cert \
   -dname "CN=<HOSTNAME>, OU=<ORGANIZATION_UNIT>
   , O=<ORGANIZATION_NAME>, \
   L=<CITY>, S=<STATE>, C=<COUNTRY_CODE>" -keypass <YOUR_PASSWORD> \
   -storetype JKS -keystore ia_demo_keystore.jks -storepass
   ```

   b. Export the certificate by running the following command:

   ```
   keytool -export -noprompt -rfc -alias ia_demo_cert \
      -file ia_demo_public_cert.cer -keystore ia_demo_keystore.jks \
      -storepass <YOUR_PASSWORD> \
    -storetype JKS
   ```

   c. Create a truststore and import the certificate into it by running the following command:

   ```
   keytool -import -noprompt -trustcacerts -alias ia_demo_cert \
      -file ia_demo_public_cert.cert -keystore ia_demo_truststore.jks \
      -storepass <YOUR_PASSWORD>
   ```

To edit the configuration files for IA Server:

1. In a text editor, open the `<IA_ROOT>/config/iaserver/application.yml` file.
2. Make sure that the following parameters are in the `application.yml` file:

   ```
   spring:
     profiles:
       include:
       - https
   ```

Caution

IA Server will not restart if there are duplicate `spring` sections in the file. By default, these sections are in alphabetical order.

3. In a text editor, open the `<IA_ROOT>/config/iaserver/application-https.yml` file.
4. Set the `clientAuth` parameter to one of the following values:
   - `none`: Disable client authentication.
   - `need`: Enable two-way TLS/SSL. The TLS/SSL client application and IA Server must perform a mutual authentication.
   - `want`: Allow the TLS/SSL client application to authenticate with IA Server, but it is not required. This option supports clients with and without certificates.
5. In the `server.ssl` section, update the following parameters to point to the keystore and truststore:

```
keyAlias: ia_demo_cert
keyPassword: <YOUR_PASSWORD>
keyStore: "file:<IA_ROOT>/config/https/ia_demo_keystore.jks"
keyStorePassword: <YOUR_PASSWORD>
keyStoreType: JKS
trustStore: "file:<IA_ROOT>/config/https/ia_demo_truststore.jks"
trustStorePassword: <YOUR_PASSWORD>
trustStoreType: JKS
```

To edit the configuration files for IA Web App:

1. In a text editor, open the `<IA_ROOT>/config/iawebapp/application.yml` file.
2. Make sure that the following parameters are in the `application.yml` file:

```
spring:
  profiles:
    include:
    - infoarchive.profile.HTTPS
```

3. In a text editor, open the `<IA_ROOT>/config/iawebapp/application-infoarchive.profile.HTTPS.yml` file.
4. In the `server.ssl` section, do one of the following:
   a. If you want to set up two-way TLS/SSL for IA Web App, then update the parameters as follows:

```
clientAuth: need
key-password: <YOUR_PASSWORD>
key-store: "<IA_ROOT>/config/https/ia_demo_keystore.jks"
key-store-password: <YOUR_PASSWORD>
keyStoreType: JKS
trust-store: "<IA_ROOT>/config/https/ia_demo_truststore.jks"
trust-store-password: <YOUR_PASSWORD>
trust-store-type: JKS
```

   b. If you want to set up one-way TLS/SSL for IA Web App, then update the parameters as follows:

```
clientAuth: want
key-password: <YOUR_PASSWORD>
key-store: "<IA_ROOT>/config/https/ia_demo_keystore.jks"
key-store-password: <YOUR_PASSWORD>
keyStoreType: JKS
trust-store: "<IA_ROOT>/config/https/ia_demo_truststore.jks"
trust-store-password: <YOUR_PASSWORD>
trust-store-type: JKS
```

5. Make sure that `zuul.routes.restapi.url` is set to the correct value for IA Server's URL.

To edit the configuration files for IA Shell:

1. In a text editor, open the `<IA_ROOT>/config/iashell/application.yml` file.
2. Make sure that the following parameters are in the `application.yml` file:

```
spring:
  profiles:
    include:
    - https
```

Caution

IA Shell will not restart if there are duplicate `spring` sections in the file. By default, these sections are in alphabetical order.

3. In a text editor, open the `<IA_ROOT>/config/iashell/application-https.yml` file.
4. Update the following settings (note the change from `http` to `https`):

```
gatewayUrl:
https://<IP_OF_IAWEBAPP>:<PORT_OF_IAWEBAPP>
 restApiUrl:
 https://<IP_OF_IASERVER>:<PORT_OF_IASERVER>/services
```

5. In the `ssl` section, do one of the following:
   a. If you want to set up two-way TLS/SSL for IA Shell, then update the parameters in this section as follows:

```
keyStore: <IA_ROOT>/config/https/ia_demo_keystore.jks
keyStorePassword: <YOUR_PASSWORD>
keyStoreType: JKS
trustStore: <IA_ROOT>/config/https/ia_demo_truststore.jks
trustStorePassword: <YOUR_PASSWORD>
trustStoreType: JKS
```

   b. If you want to set up one-way TLS/SSL for IA Shell, update the parameters in this section as follows:

```
keyStore:
keyStorePassword:
keyStoreType:
trustStore: <IA_ROOT>/config/https/ia_demo_truststore.jks
trustStorePassword: <YOUR_PASSWORD>
trustStoreType: JKS
```

To finalize the TLS/SSL setup:

1. Restart the OpenText Information Archive components using a command prompt so that you can check for any errors. For more information, see Starting the components with a command prompt.
2. In a web browser, go to `https://<IP_OF_IAWEBAPP>:<PORT_OF_IAWEBAPP>/#/login`.

   Note: Your web browser might display a warning or refuse the connection with an error (for example, `This site can't provide a secure connection`). This is because the self-signed certificate is not considered secure. The workaround is to import the `ia_demo_public_cert.cert` certificate into your web browser's list of trusted root certification authorities (also known as creating a security exception). You might also need to restart your web browser.

13.3 Configuring OTDS for authentication provider mode

If you used the OpenText Information Archive install software to set up OTDS, then many of the parameters in the tasks below have already been specified automatically in the configuration files.

Note: Be careful not to specify the same parameters more than once in the files.

If you are setting up OTDS after you have installed OpenText Information Archive, then you must follow all of the steps below.

If you want to run IA Web App as a standalone Spring Boot application on the same computer that you are running OTDS deployed to Apache Tomcat, you must make sure that their ports do not conflict. By default, they do conflict.

The following assumptions are made in the instructions below:

- OTDS is already installed and configured with the required users and groups.
- The OTDS administration URL is `otds_host:<OTDS_PORT>/otds-admin/`.
- The REST API endpoint is `otds_host:<OTDS_PORT>/otdsws/rest`.

To activate the OTDS integration in IA Web App that is a standalone Spring Boot application:

1. In IA Web App, configure the active profiles by updating the `include` property in the `<IA_ROOT>/config/iawebapp/application.yml` file, as illustrated below:

```
profiles:
  include:
  - infoarchive.gateway.profile.AUTHENTICATION_IN_MEMORY
  - infoarchive.gateway.profile.AUTHENTICATION_OTDS
  - CLIENTS
```

The use of the AUTHENTICATION_IN_MEMORY profile temporarily enables example user accounts (also known as in-memory user accounts) so that you can map the groups discovered from OTDS to the appropriate roles.

Note: If the same user accounts are in both of the profiles above, the user accounts in the AUTHENTICATION_IN_MEMORY profile take precedence while both profiles are active.

2. In a text editor, open the <IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.AUTHENTICATION_OTDS.yml file.
3. In the OTDS section, specify values for the following parameters:
   - location.host: The host name for the Apache Tomcat instance that hosts OTDS.
   - location.httpsPort: The HTTPS port for the Apache Tomcat instance that hosts OTDS.
   - location.port: The HTTP port for the Apache Tomcat instance that hosts OTDS.
   - password: The password to access OTDS.
   - username: The username to access OTDS. The default is otadmin@otds.admin. If when you installed OTDS, you specified a different name for the administrative user, then specify that name here.
4. In the OTDS.location section, depending on how you want your users to connect to OTDS, specify either the HTTP or HTTPS URL as follows. If you use HTTPS, you must make sure the truststore is configured on IA Web App to trust Apache Tomcat.
   - url: http://${OTDS.location.host}:${OTDS.location.port}/${OTDS.location.path}
   - url: https://${OTDS.location.host}:${OTDS.location.httpsPort}/${OTDS.location.path}
5. Restart IA Web App.
6. Log into the IA Web App as a user with the Administrator role.
7. Navigate to the **Administration > Groups** tab to map the groups discovered from OTDS to the appropriate roles.

   Tip: If you do not see the groups from OTDS that you expect to see, or if the groups are crossed out, then you might not have provided the correct credentials (user names and passwords) in the previous step of this procedure.
8. In IA Web App, configure the active profiles by updating the include property in the <IA_ROOT>/config/iawebapp/application.yml file to remove the AUTHENTICATION_IN_MEMORY profile, as illustrated below:

```
profiles:
  include:
  - infoarchive.gateway.profile.AUTHENTICATION_OTDS
  - CLIENTS
```

Caution

- In a production environment, it is very important that you remove the `AUTHENTICATION_IN_MEMORY` profile when you are done configuring the OTDS integration. Enabling these accounts significantly undermines any security measures implemented and easily allows attackers to use one of the example user accounts to gain unauthorized access to your system and the assets it contains. These out-of-the-box accounts are meant for demo and limited configuration purposes only, and the default password for each account is `password`. For more information about example user accounts, see Working with example user accounts.

9. Restart IA Web App.

To activate the OTDS integration in IA Web App that is deployed to Apache Tomcat:

1. Using a text editor, open the `<TOMCAT_ROOT>/webapps/infoarchive-webapp/WEB-INF/classes/application.yml` file.
2. Configure the active profiles by updating the `include` property in the file, as illustrated below:

   ```
   profiles:
     include:
     - infoarchive.gateway.profile.AUTHENTICATION_OTDS
   ```

   This tells the system that you are opting to use OTDS in authentication provider mode and activates the properties found in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.AUTHENTICATION_OTDS.yml` file.

   Note: Make sure that `-CLIENTS` does not appear under the `include` property.

3. In the `infoarchive.gateway` section, update the following parameters:

   - `port`: The HTTP port where Apache Tomcat is hosting IA Web App.
   - `contextPath`: The context path of IA Web App as deployed to Apache Tomcat. It must be terminated with a trailing `/` (slash).
   - `client.ssl.trust-store`: The fully qualified file path of the truststore into which the certificate of the Apache Tomcat instance that hosts OTDS at an `HTTPS` port has been imported. This is needed by the OAuth2 authentication endpoints in OTDS.
   - `client.ssl.trust-store-password`: The password for the truststore.

- client.ssl.trust-store-type: The type of truststore. Valid values are PKCS12, JKS, JCEKS, and BKS.

4. Using a text editor, open the <TOMCAT_ROOT>/webapps/infoarchive-webapp/WEB-INF/classes/application-infoarchive.gateway.profile.AUTHENTICATION_OTDS.yml file.
5. In the OTDS section, update the following parameters:
   - location.host: The host name for the Apache Tomcat instance that hosts OTDS.
   - location.port: The HTTP port for the Apache Tomcat instance that hosts OTDS.
   - location.httpsPort: The HTTPS port for the Apache Tomcat instance that hosts OTDS.
   - password: The password to access OTDS.
   - infoarchive.clients.gateway.clientSecret: The client secret for OAuth gateway client that was specified during the setup process.
   - infoarchive.clients.iawa.clientSecret: The client secret for the OAuth iawa client that was specified during the setup process.
6. In the OTDS.location section, depending on how you want your users to connect to OTDS, specify either the HTTP or HTTPS URL as follows. If you use HTTPS, you must make sure the truststore is configured on IA Web App to trust Apache Tomcat.
   - url: http://${OTDS.location.host}:${OTDS.location.port}/${OTDS.location.path}
   - url: https://${OTDS.location.host}:${OTDS.location.httpsPort}/${OTDS.location.path}
7. Using a text editor, open the <IA_ROOT>/config/iawebapp/application-CLIENTS.yml file.
8. Update the following parameters in this file:
   - Set the clientSecret for infoarchive.jdbc to the secret token for JDBC.
   - Set the clientSecret for infoarchive.cli to the secret token for IA Web App.
9. Restart IA Web App.

To finalize the OTDS configuration:

1. Connect to OTDS, using the username that you have specified in the OpenText Information Archive configuration files to access OTDS (for example, otadmin@otds.admin).

2. Locate the `infoarchive.iawa` client and go to the **OAuth Clients > Actions > Properties > Redirect URLs** tab.
3. Click the **Add** button.
4. Add the value of `preEstablishedRedirectUri` and click Save.

   Note: For more information, see Creating an OAuth Client (https://knowledge.opentext.com/knowledge/piroot/otds/v160402/otds-iwc/en/html/jsframe.htm?s-croauthcl).

5. Restart all of the OpenText Information Archive components except for PostgreSQL server.
6. In a web browser, access IA Web App. If you see the login screen, then the OTDS integration was successful.
7. After you have finished configuring partitions for LDAP or Active Directory, use the OTDS Web Administration Client to disable the `infoarchive.bootstrap` partition in OTDS.

13.4 Configuring OTDS for SSO mode

If you used the OpenText Information Archive install software to set up OTDS, then many of the parameters in the tasks below are automatically specified in the configuration files.

Note: Be careful not to specify the same parameters more than once in the files.

If you are setting up OTDS after you have installed OpenText Information Archive, then you must follow all of the steps below.

When you configure OpenText Information Archive for OTDS SSO integration mode, you configure the JDBC Oauth2 client (`infoarchive.jdbc`) in OTDS. In this configuration, OTDS acts as the OAuth2 authorization server and issues the JWT token.

You must also perform some configuration on the following components:

- *IA Server*: In this mode, OTDS issues JWT tokens instead of Gateway. The OTDS tokens are in a different format and IA Server needs to parse them differently, so some configuration of IA Server is necessary.
- *IA Shell*: The OAuth client representing IA Shell is defined in OTDS. Because OTDS issues the client secrets for OAuth2 clients, you must update IA Shell to work in this mode. The name of OAuth client for IA Shell is `infoarchive.cli`.

The following assumptions are made in the instructions below:

- OTDS is already installed.
- The OAuth2 authentication endpoints require the HTTPS protocol to be used. These instructions assume that Apache Tomcat has been configured to run an

HTTPS connector on port 8443, and you have configured the appropriate keys and certificates.

1. Make sure that OTDS is running.
2. When using JDBC driver with OpenText Information Archive configured in OTDS SSO mode, do the following to make sure that OTDS includes the user's groups and roles in the JWT token:
   a. In the OTDS Administration web client, go to **OAuth Clients > infoarchive.jdbc**.
   b. Click **Actions > Properties > Advanced**.
   c. In the **Default scopes** field, click the **Add** button, and then specify `otds:groups` and `otds:roles`.
   d. Click **Add** and then **Save**.
3. Using the OTDS Web Administration Client, enable the `infoarchive.bootstrap` partition in OTDS. You can also configure additional partitions for your IDPs (AD or LDAP). Make sure to add the partitions to the `infoarchive` access role. For more information, see the OTDS documentation.

To activate the OTDS integration in IA Web App that is a standalone Spring Boot application:

1. In IA Web App, configure the active profiles by updating the `include` property in the `<IA_ROOT>/config/iawebapp/application.yml` file, as illustrated below:

```
profiles:
  include:
  - infoarchive.gateway.profile.OTDS
```

   This tells the system that you are opting to use OTDS in SSO mode and activates the properties found in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.OTDS.yml` file.

   Note: Make sure that `-CLIENTS` does not appear under the `include` property.

2. In the `OTDS.location` section, depending on how you want your users to connect to OTDS, specify either the HTTP or HTTPS URL as follows. If you use HTTPS, you must make sure the truststore is configured on IA Web App to trust Apache Tomcat.
   - `url: http://${OTDS.location.host}:${OTDS.location.port}/${OTDS.location.path}`
   - `url: https://${OTDS.location.host}:${OTDS.location.httpsPort}/${OTDS.location.path}`
3. Because the `CLIENTS` profile is not enabled, OpenText Information Archive now ignores the contents of the `<IA_ROOT>/config/iawebapp/application-CLIENTS.yml` file. But you should still update the following parameters in this file so that when you upgrade OpenText Information Archive, you will not run into errors:

- Set the `clientSecret` for `infoarchive.jdbc` to the secret token for JDBC.
- Set the `clientSecret` for `infoarchive.cli` to the secret token for IA Web App.

Note: If you are using encrypted passwords, these should be the encrypted versions of these secrets.

Note: If you are using a load balancer at `LB_HOST:LB_PORT` in front of IAWA/Gateway instances, in the file `config\iawebapp\application-infoarchive.gateway.profile.OTDS.yml` make sure host and port in the two properties `logoutUrl` and `preEstablishedRedirectUri` below match `LB_HOST:LB_PORT` values for proper continued routing of access through the Load Balancer after logout/login cycle.

```
OTDS:
  infoarchive:
    clients:
     ...
      iawa:
        ...
        # logoutUrl: ${OTDS.infoarchive.gateway.protocol}://${infoarchive.gateway.host}:$
{infoarchive.gateway.port}${infoarchive.gateway.contextPath}logout
        logoutUrl: ${OTDS.infoarchive.gateway.protocol}://LB_HOST:LB_PORT$
{infoarchive.gateway.contextPath}logout
     ...
    ...
security:
  oauth2:
    client:
      ...
      # preEstablishedRedirectUri: ${OTDS.infoarchive.gateway.protocol}://$
{infoarchive.gateway.host}:${infoarchive.gateway.port}$
{infoarchive.gateway.contextPath}login
      preEstablishedRedirectUri: ${OTDS.infoarchive.gateway.protocol}://LB_HOST:LB_PORT$
{infoarchive.gateway.contextPath}login
```

To activate the OTDS integration in IA Web App that is deployed to Apache Tomcat:

1. Using a text editor, open the `<TOMCAT_ROOT>/webapps/infoarchive-webapp/WEB-INF/classes/application.yml` file.
2. Configure the active profiles by updating the `include` property in the file, as illustrated below:

   ```
   profiles:
     include:
     - infoarchive.gateway.profile.OTDS
   ```

   This tells the system that you are opting to use OTDS in SSO mode and activates the properties found in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.OTDS.yml` file.

   Note: Make sure that `-CLIENTS` does not appear under the `include` property.

3. In the `infoarchive.gateway` section, update the following parameters:

   - `port`: The HTTP port where Apache Tomcat is hosting IA Web App.

- `contextPath`: The context path of IA Web App as deployed to Apache Tomcat. It must be terminated with as trailing `/` (slash).
- `client.ssl.trust-store`: The fully qualified file path of the truststore into which the certificate of the Apache Tomcat instance that hosts OTDS at an `HTTPS` port has been imported. This is needed by the OAuth2 authentication endpoints in OTDS.
- `client.ssl.trust-store-password`: The password for the truststore.
- `client.ssl.trust-store-type`: The type of truststore. Valid values are `PKCS12`, `JKS`, `JCEKS`, and `BKS`.

4. Using a text editor, open the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.OTDS.yml` file.
5. In the `OTDS` section, update the following parameters:
   - `infoarchive.clients.gateway.clientSecret`: The client secret for Gateway from the answer provided during setup.
   - `infoarchive.clients.iawa.clientSecret`: The client secret for IA Web App from the from the answer provided during setup.
   - `location.host`: The host name for the Apache Tomcat instance that hosts OTDS.
   - `location.httpsPort`: The HTTPS port for the Apache Tomcat instance that hosts OTDS.
   - `location.port`: The HTTP port for the Apache Tomcat instance that hosts OTDS.
   - `password`: The password to access OTDS.
   - `username`: The username to access OTDS. The default is `otadmin@otds.admin`. If when you installed OTDS, you specified a different name for the administrative user, then specify that name here.
6. In the `OTDS.location` section, depending on how you want your users to connect to OTDS, specify either the HTTP or HTTPS URL as follows. If you use HTTPS, you must make sure the truststore is configured on IA Web App to trust Apache Tomcat.
   - `url: http://${OTDS.location.host}:${OTDS.location.port}/${OTDS.location.path}`
   - `url: https://${OTDS.location.host}:${OTDS.location.httpsPort}/${OTDS.location.path}`
7. Because the `CLIENTS` profile is not enabled, OpenText Information Archive ignores the contents of the `<IA_ROOT>/config/iawebapp/application-CLIENTS.yml` file. But you should still update the following properties in this file so that when you upgrade OpenText Information Archive, you will not run into errors:
   - Set the `clientSecret` for `infoarchive.jdbc` to the secret token for JDBC.

- Set the `clientSecret` for `infoarchive.cli` to the secret token for IA Web App.

8. Restart IA Web App.

Note: If you are using a load balancer at `LB_HOST:LB_PORT` in front of IA Web App and Gateway instances, in the file `config\iawebapp\application-infoarchive.gateway.profile.OTDS.yml` make sure host and port in the two properties `logoutUrl` and `preEstablishedRedirectUri` below match `LB_HOST:LB_PORT` values for proper continued routing of access through the Load Balancer after logout/login cycle.

```
OTDS:
  infoarchive:
    clients:
     ...
      iawa:
        ...
        # logoutUrl: ${OTDS.infoarchive.gateway.protocol}://${infoarchive.gateway.host}:$
{infoarchive.gateway.port}${infoarchive.gateway.contextPath}logout
        logoutUrl: ${OTDS.infoarchive.gateway.protocol}://LB_HOST:LB_PORT$
{infoarchive.gateway.contextPath}logout
     ...
    ...
security:
  oauth2:
    client:
      ...
      # preEstablishedRedirectUri: ${OTDS.infoarchive.gateway.protocol}://$
{infoarchive.gateway.host}:${infoarchive.gateway.port}$
{infoarchive.gateway.contextPath}login
      preEstablishedRedirectUri: ${OTDS.infoarchive.gateway.protocol}://LB_HOST:LB_PORT$
{infoarchive.gateway.contextPath}login
```

To activate the OTDS integration in IA Server:

1. In a text editor, open the `<IA_ROOT>/config/iaserver/application.yml` file.
2. Look for the `spring.profiles.include` section in the file, and then do one of the following:

   - If there is no `spring.profiles.include` section, add the following:

     ```
     spring:
       profiles:
         include:
         - otds
     ```

   - If there is a `spring.profiles.include` section, add the `- otds` profile to the section.

   Tip: By default, these sections are in alphabetical order.

   Caution
   IA Server will not restart if there are duplicate `spring` sections in the file.

   This enables IA Server to decode the JWT tokens that are issued by OTDS.

3. In a text editor, open the `<IA_ROOT>/config/iaserver/application-otds.yml` file. Update the following parameters:

   - `host`: The host name for the Apache Tomcat instance that hosts OTDS.
   - `port`: The port for the Apache Tomcat instance that hosts OTDS.
   - `httpsPort`: The HTTPS port for the Apache Tomcat instance that hosts OTDS.
   - `username`: The username to access OTDS.
   - `password`: The password to access OTDS.

   This also enables IA Server to persist the group-role mapping back into OTDS and the PostgreSQL database.

4. Restart IA Web App.

5. Log into the IA Web App as a user with the Administrator role.

   Navigate to the **Administration > Groups** tab to map the groups discovered from OTDS to the appropriate roles.

   Tip: If you do not see the groups from OTDS that you expect to see, then you might not have provided the correct credentials in earlier steps in this procedure.

13.5 Enabling OTIV integration

IA Server

Configure the following profile in OpenText Information Archive: `<IA_ROOT>/config/iaserver/application.yml`:

```
:
:
spring:
    cloud:
        :
        :
    profiles:
        include:
        - otds
        - otiv
:
:
```

Configure the following ITIV user (OAuth2 client) in `<IA_ROOT>/config/iaserver/application-otiv.yml`:

```
:

otiv:
    username: iv-publisher@OAuthClients
```

IA Web App/Gateway

Add support to the Gateway configuration to enable integration with OTIV using the Spring profile (`infoarchive.gateway.profile.otiv`) in `<IA_ROOT>/config/iawebapp/application.yml`:

```
:
:
spring:
    application:
        name: infoarchive.gateway
:
    profiles:
        include:
        - infoarchive.gateway.profile.OTDS
        - infoarchive.gateway.profile.OTIV
```

OpenText Information Archive must be configured in OTDS SSO mode (for instance, `infoarchive.gateway.profile.otds` is active). A mechanism is needed in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.OTIV.yml` file to specify the location of OTIV services as shown below:

```
otiv:
    viewer:
    # URL of the OTIV Viewer service
    url: "https://10.1.2.3"
    publication:
    # URL of the OTIV Publication service
    url: "https://10.1.2.3"
    search:
    # URL of the OTIV Search service
    url: "https://10.1.2.3"
```

IntelligentView_MSI.properties

Update the IntelligentView_MSI.properties file:

1. Set the `HTTP_PROTOCOL` to `https`.
2. Set the keystore path and password. The alias can be left blank:

```
HTTP_PROTOCOL=https
#If https is enabled, provide the file path and password for the java keystore.
KEYSTORE_PATH=C:\IA\keystore.jks
KEYSTORE_PASSWORD=<password>
KEYSTORE_ALIAS=

DEFAULT_HOST=10.1.2.3

DEFAULT_DB_HOST=10.1.2.3
DEFAULT_DB_PORT=5432
DEFAULT_DB_NAME=iv
```

3. Set the OTDS section. For `OTDS_ORIGIN`, it is recommended that you use `https` and the `https` port, but it is possible to use `http` and the regular port (http://10.1.2.3:8090).

```
OTDS_ORIGIN=https://10.1.2.3:8443/otdsws
#Providing OTDS_ADMIN_USER will trigger the installer to configure OTDS.
#If you ran OTDSConfig manually, you should leave OTDS_ADMIN_USER blank and
instead
#provide the OTDSConfig output (properties file) to the installer property
OTDS_AUTH_FILE_PATH.
#Alternatively, you can merge the OTDSConfig properties file into this one.
OTDS_ADMIN_USER=otadmin@otds.admin
#OTDS_ADMIN_PW is required if OTDS_ADMIN_USER is provided.
OTDS_ADMIN_PW=<password>
```

13.6 Secure Sockets Layer (SSL)

13.6.1 Manually setting up SSL for a component

The following steps outline the process of setting up TLS/SSL for OpenText Information Archive at a high level for three different types of configurations:

- A demo configuration
- A production configuration where OpenText Information Archive components share a keystore and truststore
- A production configuration where OpenText Information Archive components use separate keystores and truststores

Subsequent topics explain in greater detail how to set up TLS/SSL for these different types of configurations.

If you are not experienced in setting up TLS/SSL for enterprise software, you should start by setting up TLS/SSL for a demo configuration. This is the simplest way to set up TLS/SSL for OpenText Information Archive, so you can use it to learn about TLS/SSL. The most complicated setup is a production configuration with separate keystores and truststores.

Notes

- Some of the code samples shown in this guide contain \ newline characters that allow the code to flow over several lines for readability and formatting purposes. The \ newline characters are not to be used as part of the code shown, and you must manually remove them prior to using the code.
- When you specify parameters in a configuration file, be careful not to include any extra spaces at the end. An extra space might cause the value to be interpreted incorrectly.

To set up TLS/SSL in a demo configuration:

1. Create a self-signed certificate and keystore for all the OpenText Information Archive components.
2. Export the certificate.
3. Create a truststore and import the certificate.
4. Edit configuration files for IA Server, IA Web App, and IA Shell.
5. Update the OpenText Information Archive applications that you want to use.
6. Start the TLS/SSL versions of the components.

To set up TLS/SSL in a production configuration with a shared keystore and truststore:

1. Obtain signed certificates for IA Server, IA Web App, PostgreSQL, and IA Shell.

2. Import each of the certificates back into the keystore so that the signed certificate is installed.
3. Import each of the certificates into the truststore.
4. Edit configuration files for IA Server, IA Web App, PostgreSQL server, and IA Shell.
5. Update the OpenText Information Archive applications that you want to use.
6. Start the TLS/SSL versions of the components.

To set up TLS/SSL in a production configuration with separate keystores and truststores:

1. Obtain signed certificates for IA Server, IA Web App, PostgreSQL server, and IA Shell.
2. Import each component's certificate back into its own keystore so that the signed certificate is installed.
3. Import each component's certificate into each of the other component's truststores.
4. Edit configuration files for IA Server, IA Web App, PostgreSQL server, and IA Shell.
5. Update the OpenText Information Archive applications that you want to use.
6. Start the TLS/SSL versions of the components.

13.6.1.1 Setting up TLS/SSL in a production configuration with a shared keystore and truststore

In a production configuration, OpenText Information Archive components are distributed over separate computers. The following assumptions are made in the instructions below:

- You installed a production configuration.
- You are using a shared keystore and truststore on a network location.
- You want your keystore and truststore to be in `PKCS12` format. You can use `JKS`, `JCEKS`, or `PKCS11` instead, if you want. The instructions for setting up TLS/SSL in a demo configuration use `JKS`, so see those instructions for the necessary settings.
- You are using a Windows computer and Java Keytool to manage your keys and certificates. However, you can use a Linux computer and other software to manage your keys and certificates.
- You added the directory that contains the Keytool executable to your PATH system variable.
- When you installed the system components, you chose to install the components as system services using SSL.

- The aliases for each certificate are as follows:
  - IA Server: `ia_server_cert`
  - IA Web App: `ia_webapp_cert`
  - IA Shell: `ia_shell_cert`

  However, the aliases can be whatever you want. If you do not use the aliases above, then when you follow the procedures below, make sure that you use your aliases consistently instead.

Note: For an example of how to obtain signed certificates and import them into keystores and truststores, see Example: obtaining and importing signed certificates for TLS/SSL.

To prepare the certificates, keystore, and truststore:

1. Obtain signed certificates for IA Server, IA Web App, PostgreSQL server, and IA Shell.
2. Stop the OpenText Information Archive system services.
3. Copy the signed certificates into the directory where the keystore is located.
4. In a command prompt, go to the directory where the keystore is located.
5. Import the certificates back into the keystore so that the signed certificates are installed by doing the following:
   a. Import the IA Server certificate back into the keystore by running the following command:

```
keytool -import -noprompt -alias ia_server_cert \
   -file <FILE_NAME> -keystore <KEYSTORE_NAME> \
   -storepass <KEYSTORE_PASSWORD> -storetype PKCS12
```

   b. Import the IA Web App certificate back into the keystore by running the following command:

```
keytool -import -noprompt -alias ia_webapp_cert \
   -file <FILE_NAME> -keystore <KEYSTORE_NAME> \
   -storepass <KEYSTORE_PASSWORD> -storetype PKCS12
```

   c. Import the IA Shell certificate back into the keystore by running the following command:

```
keytool -import -noprompt -alias ia_shell_cert \
   -file <FILE_NAME> -keystore <KEYSTORE_NAME> \
   -storepass <KEYSTORE_PASSWORD> -storetype PKCS12
```

6. Copy the certificates into the directory where the truststore is located. It is recommended to put this in the `<IA_SHARED_RESOURCES>` directory.
7. In a command prompt, go to the directory where the truststore is located.
8. Import the certificates into the truststore by doing the following:

a. Import the IA Server certificate into the truststore by running the following command:

```
keytool -import -noprompt -alias ia_server_cert \
   -file <FILE_NAME> -keystore <TRUSTSTORE_NAME> \
   -storepass <TRUSTSTORE_PASSWORD> -storetype PKCS12
```

b. Import the IA Web App certificate into the truststore by running the following command:

```
keytool -import -noprompt -alias ia_webapp_cert \
   -file <FILE_NAME> -keystore <TRUSTSTORE_NAME> \
   -storepass <TRUSTSTORE_PASSWORD> -storetype PKCS12
```

c. Import the IA Shell certificate into the truststore by running the following command:

```
keytool -import -noprompt -alias ia_shell_cert \
   -file <FILE_NAME> -keystore <TRUSTSTORE_NAME> \
   -storepass <TRUSTSTORE_PASSWORD> -storetype PKCS12
```

To edit the configuration files for IA Server:

- On each computer that hosts IA Server, edit the configuration files for IA Server by doing the following:

a. In a text editor, open the `<IA_ROOT>/config/iaserver/application.yml` file.

b. Make sure that the following parameters are in the `application.yml` file:

```
spring:
  profiles:
    include:
    - https
```

Caution

IA Server will not restart if there are duplicate `spring` sections in the file. By default, these sections are in alphabetical order.

c. In a text editor, open the `<IA_ROOT>/config/iaserver/application-https.yml` file.

d. Set the `clientAuth` parameter to one of the following values:

- `none`: Disable client authentication.
- `need`: Enable two-way TLS/SSL. The TLS/SSL client application and IA Server must perform a mutual authentication.
- `want`: Allow the TLS/SSL client application to authenticate with IA Server, but it is not required. This option supports clients with and without certificates.

e. In the `server.ssl` section, update the following parameters to point to the keystore and truststore:

```
keyAlias: ia_server_cert
keyPassword: <PRIVATE_KEY_PASSWORD>
keyStore: "file:<KEYSTORE_PATH_AND_FILENAME>"
keyStorePassword: <KEYSTORE_PASSWORD>
```

```
keyStoreType: PKCS12
trustStore: "file:<TRUSTSTORE_PATH_AND_FILENAME>"
trustStorePassword: <TRUSTSTORE_PASSWORD>
trustStoreType: PKCS12
```

To edit the configuration files for IA Web App:

- On each computer that hosts IA Web App, edit the configuration files for IA Web App by doing the following:

  a. In a text editor, open the `<IA_ROOT>/config/iawebapp/application.yml` file.

  b. Make sure that the following parameters are in the `application.yml` file:

```
spring:
  profiles:
    include:
    - infoarchive.profile.HTTPS
```

> **Caution**
> IA Web App will not restart if there are duplicate `spring` sections in the file. By default, these sections are in alphabetical order.

  c. In a text editor, open the `<IA_ROOT>/config/iawebapp/application-infoarchive.profile.HTTPS.yml` file.

  d. In the `server.ssl` section, do one of the following:

  i. If you want to set up two-way TLS/SSL for IA Web App, then update the parameters as follows:

```
clientAuth: need
key-password: <SECRET_KEY_PASSWORD>
key-store: "<KEYSTORE_PATH_AND_FILENAME>"
key-store-password: <KEYSTORE_PASSWORD>
keyStoreType: PKCS12
trust-store: "<TRUSTSTORE_PATH_AND_FILENAME>"
trust-store-password: <TRUSTSTORE_PASSWORD>
trust-store-type: PKCS12
```

  ii. If you want to set up one-way TLS/SSL for IA Web App, then update the parameters as follows:

```
clientAuth: want
key-password: <SECRET_KEY_PASSWORD>
key-store: "<KEYSTORE_PATH_AND_FILENAME>"
key-store-password: <KEYSTORE_PASSWORD>
keyStoreType: PKCS12
trust-store: "<TRUSTSTORE_PATH_AND_FILENAME>"
trust-store-password: <TRUSTSTORE_PASSWORD>
trust-store-type: PKCS12
```

> **Note:** If you have enabled password encryption, then you should specify the encrypted versions of the passwords above.

  e. Make sure that the `zuul.routes.restapi.url` parameter is set to the correct value for IA Server's URL.

To edit the configuration files for IA Shell:

- On each computer that hosts IA Shell, edit the configuration files for IA Shell by doing the following:

  a. In a text editor, open the `<IA_ROOT>/config/iashell/application.yml` file.

  b. Make sure that the following parameters are in the `application.yml` file:

```
spring:
  profiles:
    include:
    - https
```

Caution

IA Shell will not restart if there are duplicate `spring` sections in the file. By default, these sections are in alphabetical order.

  c. In a text editor, open the `<IA_ROOT>/config/iashell/application-https.yml` file.

  d. Update the following settings (note the change from `http` to `https`):

```
gatewayURL: https://<IP_OF_IAWEBAPP>:<PORT_NUMBER>
restApiUrl: https://<IP_OF_IASERVER>:<PORT_NUMBER>/services
```

  e. In the `ssl` section, do one of the following:

    i. If you want to set up two-way TLS/SSL for IA Shell, then update the parameters in this section as follows:

```
keyStore: <KEYSTORE_PATH_AND_FILENAME>
keyStorePassword: <KEYSTORE_PASSWORD>
keyStoreType: PKCS12
trustStore: <TRUSTSTORE_PATH_AND_FILENAME>
trustStorePassword: <TRUSTSTORE_PASSWORD>
trustStoreType: PKCS12
```

    ii. If you want to set up one-way TLS/SSL for IA Shell, then update the parameters in this section as follows:

```
keyStore:
keyStorePassword:
keyStoreType:
trustStore: <TRUSTSTORE_PATH_AND_FILENAME>
trustStorePassword: <TRUSTSTORE_PASSWORD>
trustStoreType: PKCS12
```

To finalize the TLS/SSL setup:

1. Restart the OpenText Information Archive components using a command prompt so that you can check for any errors. For more information, see Starting the components with a command prompt.

2. Restart the OpenText Information Archive system services.

3. In a web browser, go to `https://<IP_OF_IAWEBAPP>:<PORT_OF_IAWEBAPP>/#/login`.

Note: Your web browser might display a warning or refuse the connection with an error (for example, `This site can't provide a secure`

connection). This is because the self-signed certificate is not considered secure. The workaround is to import the ia_demo_public_cert.cer certificate into your web browser's list of trusted root certification authorities (also known as creating a security exception). You might also need to restart your web browser.

13.6.2 Excluding weaker cipher suites

To exclude some of the weaker cipher suites and TLS protocols for IA Server, you need to edit the <IA_ROOT>/config/iaserver/application-https.yml file to add some parameters in the server.ssl section. Below is an example of what the file looks like after it has been edited.

```
key-store: "file:config/iawebapp/https/webappKeystore.p12"
key-store-password: 6LOQa+4hLghmPwo/R7UR3ekQ97oPxR/65tyPzd6v9ZA=
key-password: 6LOQa+4hLghmPwo/R7UR3ekQ97oPxR/65tyPzd6v9ZA=
keyStoreType: PKCS12
trust-store: "file:config/iawebapp/https/webappTruststore.p12"
trust-store-password: 6LOQa+4hLghmPwo/R7UR3ekQ97oPxR/65tyPzd6v9ZA=
trust-store-type: PKCS12
ciphers: ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-
SHA
protocol: TLS
enabled-protocols: TLSv1.2
```

IA Web App supports the same set of TLS protocols and ciphers to the ones supported by IA Server. To exclude some of the weaker cipher suites and TLS protocols for IA Web App, you need to edit the <IA_ROOT>/config/iawebapp/application-infoarchive.profile.HTTPS.yml file to add some parameters in the server.ssl section. Below is an example of what the file looks like after it has been edited:

```
keyStore: "config/iaserver/https/iaserverKeystore.p12"
keyStorePassword: 6LOQa+4hLghmPwo/R7UR3ekQ97oPxR/65tyPzd6v9ZA=
keyPassword: 6LOQa+4hLghmPwo/R7UR3ekQ97oPxR/65tyPzd6v9ZA=
keyStoreType: PKCS12
keyAlias: ia_server_cert

trustStore: "config/iaserver/https/iaserverTruststore.jks"
trustStorePassword: 6LOQa+4hLghmPwo/R7UR3ekQ97oPxR/65tyPzd6v9ZA=
trustStoreType: JKS
clientAuth: need
ciphers: ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-
SHA
protocol: TLS
enabled-protocols: TLSv1.2
```

Note: The added parameters must line up with the other parameters as they do in the examples above.

To specify TLS protocols and cipher suites for IA Server or IA Web App:

1. Do one of the following:

    a. To specify TLS protocols and cipher suites for IA Server, in a text editor, open the <IA_ROOT>/config/iaserver/application-https.yml file.

b. To specify TLS protocols and cipher suites for IA Web App, in a text editor, open the `<IA_ROOT>/config/iawebapp/application.yml` file.

2. Add the following parameters to the `server.ssl` section of the file, after the `clientAuth` parameter:

   - `ciphers`
   - `protocol`
   - `enabled-protocols`

3. Specify the ciphers that you want to support with the `ciphers` parameter. For example:

```
ciphers: ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-SHA
```

4. Specify the protocol that you want to support with the `protocol` parameter. For example:

```
protocol: TLS
```

5. Specify the version of the protocol that you want to support with the `enabled-protocols` parameter. For example:

```
enabled-protocols: TLSv1.2
```

6. Restart the component that you are specifying TLS protocols and cipher suites for (either IA Server or IA Web App).

13.6.3 Example: obtaining and importing signed certificates for TLS/SSL

As part of setting up TLS/SSL in a production configuration, you obtain signed digital certificates from a CA, and then import the certificate into keystores. For more information about setting up TLS/SSL in a production configuration, as well as related topics such as certificates and keystores, see Protecting data in transit.

To obtain a signed certificate, you create a Certificate Signing Request (CSR) and submit it to a CA. The CSR includes encoded text that the CA requires to issue a certificate. This text includes identifying information about who is requesting the certificate, including the legal name of your organization and location, as well as the public key for the certificate. You usually create a corresponding private key at the same time, which you keep secret.

Depending on the CA, you might create the CSR in one of several different formats, using one of several different tools, and your keystore and truststore can be in different formats. This topic is an example of one way to create a CSR and import the signed certificate into a keystore. It describes how to use Java Keytool to create a CSR in the Base-64 encoded PEM format, with keystores and truststores in PCKS format.

Note: You must use the same alias throughout these steps.

1. The following command generates a keypair and a certificate. This certificate is not yet signed, so while you can use it for a demo configuration, you should not use it in a production configuration until it is signed because web browsers will not trust it:

```
keytool -genkey -noprompt -alias <CERTIFICATE_ALIAS> -keyalg RSA \
   -keysize 2048 -dname "CN=<SERVER_FQDN>, OU=<ORGANIZATION_UNIT>, \
   O=<ORGANIZATION_NAME>, L=<CITY>, S=<STATE>, C=<COUNTRY_CODE>" \
   -keypass <PRIVATE_KEY_PASSWORD> -storetype PKCS12 -keystore
serverKeystore.p12 \
   -storepass <KEYSTORE_PASSWORD>
```

2. The following command creates a CSR:

```
keytool -certreq -alias <CERTIFICATE_ALIAS> -keyalg RSA -file
<CSR_FILENAME> \
-keystore serverKeystore.p12 -storepass <KEYSTORE_PASSWORD> \
-ext 'san=dns:localhost'
```

   Chrome trusts a certificate only if a Subject Alternative Name (SAN) is specified that matches with what is used in the browser to connect to the IA Web App. Refer to the Oracle Java SE Documentation to learn how to use the -ext option to specify more than one SAN (hostname, IP, *etc*).

3. Send the CSR file to the CA.

   The CA sends one or more certificate files back to you.

4. If the CA sends multiple PEM files to you, you must concatenate them into the right order. The order must start with the root, then any intermediary, then lastly your domain.

   For example, if the CA sends you three PEM files (`ca-chain.cert.pem`, `intermediary.cert.pem`, and `localhost.pem`), you can concatenate them by using the following command:

```
cat ca-chain.cert.pem intermediary.cert.pem localhost.pem > bundle.pem
```

5. Import the `bundle.pem` file into the keystore so that the signed certificate is installed:

```
keytool -import -trustcacerts -alias <CERTIFICATE_ALIAS> -file bundle.pem \
   -keystore serverKeystore.p12 -storepass <KEYSTORE_PASSWORD>
```

13.7 About the OpenText Information Archive install software

The OpenText Information Archive installation software consists of a number of scripts and a Java application. This appendix describes these files as well as various ways to customize the installation process to tweak how the configuration files are generated.

13.7.1 Scripts

The installation software consists of the following scripts in the `<IA_ROOT>` directory structure:

- `install(.bat)`:
  - Top-level installation script that simplifies installing a demo environment, upgrading from a previous version, or installing a new current production configuration.
  - Asks questions to determine what to do and then invokes the `bin/setup` script.
  - Intended only for manual invocation. It does not support any command line parameters and is not well suited to be invoked programmatically or be scripted against.
- `bin/setup(.bat)`:
  - Main installation script that performs a Java version check, extracts the database distribution, and copies the `webapp *.jar` file to the equivalent `*.war` file to allow deployment into external Apache Tomcat.
  - Attempts to correct file and directory permissions, and invokes the `bin/iasetup` script and/or the `bin/install_services` script.
  - Supports only a few explicitly validated command line parameters and therefore provides a very streamlined experience with only limited options for customization. It can be scripted against, in order to partially automate the installation process, but it is likely not flexible enough to be solely relied upon.
  - Generally only needed to be invoked once per deployment, after which the `bin/iasetup` script should be used for fine-tuning and reconfiguration instead.
- `bin/iasetup(.bat)`:
  - Bootstrap script for the `iasetup` Java application that generates the configuration files for all system components.
  - Delegates all specified parameters directly to the `iasetup` Java application, and therefore supports many command-line parameters and allows considerable tweaking for installation.

  - Can easily be scripted against and used to automate the installation process, with a lot of flexibility to control specific details of the process.

- `bin/install_services(.bat)`:

  - Script used to install system services for the various OpenText Information Archive components.
  - Can be invoked interactively, in which case questions will be asked with regards to which services should be installed or uninstalled.
  - Can also be invoked with command-line parameters to provide predefined answers to these questions and automate the installation as much as possible.

- `bin/install_service` (Linux only):

  - Script only available for Linux deployments and can be used for individual service installation.
  - The same results can be accomplished by passing the appropriate command-line parameters to the `bin/install-services` script, but for the implementation of the Linux Shell scripts, it was more convenient to do this with a separate script than it was for the Windows batch script.

13.7.2 iasetup

The `iasetup` Java application provides flexibility by means of two separate mechanisms, allowing a lot of freedom to automate its invocation in almost any way needed. It is implemented as a Spring Boot application that defines a specific configuration file in YML format, where each individual parameter can be independently overridden in multiple ways.

The main configuration file for `iasetup` is the `<IA_ROOT>/config/iasetup/application.yml` file. This file contains a few default values, but the majority of them are left blank and will either have to be provided interactively by means of active manual input or passively beforehand.

In general, because `iasetup` is a Spring Boot application, any way Spring Boot allows you to override configuration applies here as well, but two Spring-based mechanisms will be explicitly documented here because the tool has been designed, implemented, and tested with primarily these two mechanisms in mind. The `iasetup` application also relies on loading an existing configuration, but that is implemented explicitly by `iasetup` and is not an out-of-the-box Spring feature.

1. Predefined configuration templates

   - By defining Spring profiles, it is possible to load profile-specific configuration files. It is even possible to mix and match multiple profiles, though care must be taken with regards to overlapping but conflicting configuration settings between actively applied profiles. For `iasetup`, the profile-specific configuration files are looked for in the same directory as the main configuration file mentioned above.

2. Command-line parameters

    - Spring Boot interprets any command-line parameter that starts with a double dash (`--`) as an explicit override of a configuration setting. For the most part, Spring will validate the name so that it only accepts parameters that it expects, and it will end when provided with a parameter that it does not expect, to avoid ignoring accidental typos or, for other reasons, an unknown configuration that was explicitly specified and expected to be used.

3. Silently inheriting existing configuration

    - This is not a Spring mechanism, but specific to `iasetup`. However, it does have a large enough impact to warrant being mentioned explicitly here as well.
    - When running `iasetup` for an already configured environment, it will attempt to load the existing configuration for those settings that have not yet been specified or overridden by means of the above two mechanisms.

The `iasetup` script supports these two Spring-based mechanisms by interpreting any command-line parameter that does not start with a double dash as a profile to be enabled, while letting Spring Boot simply interpret all remaining command-line parameters prefixed with such a double dash as individual configuration setting overrides.

Enabling profiles for which there is no corresponding configuration file has no effect, and there will be no warning either, because this logic is simply delegated to Spring Boot, which does not require any profile-specific configuration to exist.

This does mean that typos in to-be-enabled profiles will not be prevented, so care must be taken to specify the appropriate profile name when relying on this mechanism.

13.7.3 Silent automation

`iasetup` can be run in either interactive or in non-interactive mode.

When run in interactive mode, questions will be asked for each setting for which no value has been specified beforehand. When running in non-interactive mode, all applicable settings must have a value specified, or an error will be generated and the process will end before generating the configuration files.

In order to silently automate, it will be necessary to rely on the non-interactive mode to avoid ending up with the logic waiting indefinitely for input that is never going to be provided, in case the predefined configuration is missing one or more settings.

13.7.4 Profile-specific configuration templates

When running `iasetup`, one of the questions near the end is whether to save the collected information as a new configuration template. If answered affirmatively, a profile-specific configuration template will be persisted in the directory where all profile configuration files are stored. In case a file with the same name already exists, it will be silently overwritten after being backed up automatically.

This mechanism can be used to use `iasetup` interactively and generate a configuration template that can be applied silently to one or more deployments at a later point in time. It can also act as a form of documentation to later identify how a certain environment was configured by `iasetup` and/or reset the configuration to the same state, in case certain changes were made (manually or automatically) that need to be reverted, for example.

For convenience purposes, OpenText Information Archive ships with a number of such configuration templates:

- `application-applications.yml`
  - Only generates the configuration for applications
  - Invoked by means of `<IA_ROOT>/bin/iasetup applications`
- `application-decrypt-passwords.yml`
  - Decrypts passwords
  - Invoked by means of `<IA_ROOT>/bin/iasetup decrypt-passwords`
- `application-demo.yml`
  - Generates configuration for a demo deployment
  - Invoked by means of `<IA_ROOT>/bin/iasetup demo`
- `application-encrypt-passwords.yml`
  - Encrypts passwords for an existing deployment
  - Invoked by means of `<IA_ROOT>/bin/iasetup encrypt-passwords`
- `application-iaserver.yml`
  - Only generates configuration for IA Server
  - Invoked by means of `<IA_ROOT>/bin/iasetup iaserver`
- `application-iashell.yml`
  - Only generates configuration for IA Shell
  - Invoked by means of `<IA_ROOT>/bin/iasetup iashell`
- `application-iawebapp.yml`
  - Only generates configuration for IA Web App

  - Invoked by means of `<IA_ROOT>/bin/iasetup iawebapp`

- `application-init.yml`
  - Generates an `iasetup` configuration template for the current existing configuration
  - Invoked by means of `<IA_ROOT>/bin/iasetup init`
- `application-new.yml`
  - Disables the option to upgrade from a previous installation
  - Invoked by means of `<IA_ROOT>/bin/iasetup new`
- `application-password-encryption.yml`
  - Only generates the configuration for the password encryption tool
  - Invoked by means of `<IA_ROOT>/bin/iasetup password-encryption`
- `application-upgrade.yml`
  - Forces an upgrade from a previous installation
  - Invoked by means of `<IA_ROOT>/bin/iasetup upgrade`

Some of these templates can be combined in a single run of `iasetup`, while others are mutually exclusive. For example, the templates that single out a specific component (`applications`, `iaserver`, `iawebapp`, and `iashell`) are all mutually exclusive, but they can be combined with a few others. For example:

- `iasetup upgrade iawebapp`
- `iasetup encrypt-passwords iashell`
- `iasetup decrypt-passwords iaserver`
- `iasetup init password-encryption`

It is also possible to copy one of these templates and make the necessary modifications to allow combining them. For example, combining `iawebapp` and `iashell` in a single installation can be done by merging the settings in those two files and calling it something like `application-iawebapp_and_iashell.yml`. The same applies to any persisted configuration templates generated by running `iasetup` beforehand. Any of the persisted settings can be modified and tweaked to accomplish the desired results.

It is also possible to override individual settings in the enabled profile specific configuration templates by using the double dash prefixed parameters on the command line. When installing a demo environment, for example, password encryption is normally not enabled (but is already configured for convenience purposes). You can run any of the following commands to deploy a demo environment with encrypted passwords:

- `iasetup demo encrypt-passwords`

- `iasetup demo --passwordEncryption.enabled=true`
- By means of two separate commands:
  - `iasetup demo`
  - `iasetup encrypt-passwords`

It is not feasible to document each individual `iasetup` configuration setting individually, but they are all present in the main `iasetup` configuration file, `<IA_ROOT>/config/iasetup/application.yml`. You just need to convert the hierarchical YML structure to a property notation, by replacing each indented level with a dot (`.`).

For example:

```
applicationSettings:
  defaultConfig:
    structuredData:
      database:
        admin:
          password: secret123
```

becomes the following:

```
applicationSettings.defaultConfig.structuredData.database.admin.password=secret123
```

Most of the setup configuration fields closely resemble or are even outright identical to the configuration settings used by the generated configuration files, but those that do not should be named or structured intuitively enough to make it reasonably clear how they affect the generated configuration and/or `iasetup` behavior.

Some experimentation with running `iasetup` interactively and inspecting the persisted configuration template might also help clarify things that may be less obvious initially.

13.8 OpenText Information Archive setup – Interview questions

This section describes the current setup questions. If no additional clarification is needed, the question is simply listed with an explanation of `N/A`.

For all passwords and sensitive information, unless an automatically generated password is used, the value needs to be entered twice.

The following indicates the possible sections that you are asked about during the setup process:

- Setup configuration settings
- PostgreSQL Server configuration settings
- OTDS Initialization configuration settings
- Information Archive Server configuration settings

- Optional Information Archive Server profiles
- Information Archive Gateway configuration settings
- Optional Information Archive Gateway profiles
- Information Archive Shell configuration settings
- Application default configuration settings
- SAP Connector configuration settings

13.8.1 General questions asked when configuring any component

`Install Information Archive using a demo configuration? (Y/n)`

- If you say Yes, the system configures all the required components on one machine and uses the basic options such as in memory accounts, no SSL, no encryption of configuration files.
- If you say Yes, this configuration must not be used as a basis for a production deployment.

`Shared resources location:`

- This specifies a location outside of the installation directory to be used with default answers when setup needs to ask for specific file/directory locations.
- By ensuring any data referenced by configuration files is stored outside of the installation directory, a new installation folder can be used without changing paths in the files in the `config` directory during upgrade /patch installation and data will not be accidentally removed together with no longer used installation folders.

`Generate passwords: (y/N)`

- Convenience mechanism if you want the system to generate passwords. It is always possible to explicitly specify a password, but enabling the generation of passwords allows the setup program to generate a random one, where applicable.
- This can result in unsupported passwords due to specific characters that cause problems. Not recommended for production use cases unless the passwords are reviewed afterwards.

`Display auto-generated passwords: (y/N)`

- Normally passwords are never shown but, when encrypting passwords, there would be no way to know the generated passwords. Displaying them in the console output becomes the only way to know what the passwords are.
- Only applicable when password generation has been enabled.

```
Use Spring Cloud Vault to store configuration secrets: (y/N)
```

- Spring Vault configuration is currently not generated by setup and needs to be provided manually.
- If you plan to use Spring Vault, however, all passwords/secrets need to be stored in Vault instead of the configuration files. Answering yes to this question only enables the profile for the components to use Vault and extracts all passwords to dedicated Vault JSON files.

13.8.2 Questions about encryption of passwords

```
Enable encryption of passwords: (y/N)
```
This is specific to passwords in configuration files. Currently only available when not using Spring Cloud Vault. Passwords also include secret keys.

```
Password encryption security provider {1=SUN_JCE,
2=BOUNCY_CASTLE, 3=GEMALTO}: (1)
```

- If you want to use Gemalto, you need to ensure that the Gemalto libraries are in the classpath of the setup program; otherwise, setup will end (see section 7 “Encrypting component passwords and secret tokens in configuration files” in *OpenText Information Archive - Encryption Guide (EARCORE-AGE)*) and section 7.2.3 “Using the password encryption utility to manually encrypt passwords and tokens” in *OpenText Information Archive - Encryption Guide (EARCORE-AGE)*).
- Each component configured with setup during the same run requires the Gemalto jars on the classpath to decrypt the passwords again.

13.8.2.1 Password generation questions

The system asks these questions if you opted to generate passwords

```
Generated password Entropy: (40.0)
```
Used to ensure a certain complexity for the generated password.

```
Generated password minimum length ( <= 32 ): (8)
```
N/A

```
Generated password character types {1=ALPHANUMERICSYMBOL,
2=ALPHANUMERIC, 3=ALPHA, 4=NUMERIC}: (1)
```

- Used to ensure the generated password conforms to the type requested. For example, the default ensures the password has at least one alpha, one numeric, and one symbol.
- Not all components properly support all symbols. It might be necessary to fallback to ALPHANUMERIC passwords only or update the generated passwords afterwards.

13.8.3 Finalizing the configuration setting

`Save this setup configuration as a template: (y/N)`
Choose Y if you want to potentially review the setup template.

`Generate transcript: (y/N)`
The transcript saves all console output, including provided answers to a text file so that it can be reviewed afterwards in case of unexpected problems.

`Write transcript to:`
If the transcript is to be generated, this is the location where it will be written to.

`Include specified passwords in transcript: (y/N)`

- Passwords are sensitive information and normally masked in the console. Including any provided passwords in the transcript would potentially compromise them and is therefore disabled by default, but can be optionally enabled.
- This includes both manually typed in passwords as well as already previously provided passwords (in case of review), but excludes generated passwords.
- In order to include generated passwords in the transcript, the option to display them must have been enabled, which prints them as console output and implicitly includes them in the transcript, regardless of whether specified passwords are included.

`Initialize configuration (generate actual configuration files): (Y/n)`

- Whether to actually generate the configuration files or not.
- It is possible to merely generate a template to be replayed at a later point in time on the same or a different machine/location in which case you might not want to generate the actual config files immediately.

13.8.4 PSQL system data configuration questions

`Configure PostgreSQL Server: (Y/n)`

- As a convenience, setup can generate basic configuration files for PostgreSQL Server.
- The resulting configuration however is not production ready and almost always needs to be reviewed and updated afterwards before startup.
- For production and most other deployments it is recommended to install and configure PostgreSQL yourself and not rely on OpenText Information Archive setup to ensure it is secure and properly configured for the environment in which it is to be deployed.

`Enable PostgreSQL Server full same network client access: (Y/n)`

- OOTB PostgreSQL Server only allows localhost access, which means it is not possible to connect to PostgreSQL running on a separate machine.
- As a convenience, setup can configure PostgreSQL to allow access from the same network so PostgreSQL can be connected to from anywhere on the same/local network.
- For production it highly depends on network configuration and security considerations/firewalls etc, which connections should be allowed/accepted which requires more in-depth analysis and (manual) configuration than setup can provide.

`PostgreSQL Server listen address:`

- The IP address PostgreSQL Server needs to listen for/accept connections on. This primarily affects multi-home machines and affects through which networks Postgres can be connected to.
- A listen address of `127.x.x.x` for example would only allow connections from the same machine.
- A listen address of `0.0.0.0` on the other hand would listen for connections on all IP's of the machine.
- Consult official PostgreSQL documentation for more information.

`PostgreSQL Server port (<= 65536 ): (5432)`

- The TCP/IP port PostgreSQL Server should accept connections on.
- Combined with the `jdbc:postgresql://` protocol and listen address, this would be the address clients need to connect to.

13.8.4.1 Configuring cluster questions

Create PostgreSQL Server database cluster: (Y/n)

- As a convenience, setup can create a pre–configured empty PostgreSQL datanode. However, this is only suited for basic scenario's and should be avoided in production deployments.
- There are specific datanode settings that cannot be changed after creation that can affect fundamental PostgreSQL behavior.
- For production use cases, it is strongly recommended to not leave this up to setup which only exposes limited control options and instead take full control over PostgreSQL datanode creation by manually creating it.
- Please refer to official PostgreSQL documentation for more details.

PostgreSQL Server superuser username: (postgres)

- The PostgreSQL superuser/datanode user name.

- Not persisted in `config` files, only used during datanode creation
- This account will have full permission to do anything with PostgreSQL

PostgreSQL Server superuser password:

- The password for the PostgreSQL superuser/datanode user
- Not persisted in `config` files, only used during datanode creation

PostgreSQL Server database cluster data directory:

- The directory to create the PostgreSQL datanode in.

13.8.4.2 SSL for PostgreSQL Server

Use SSL for PostgreSQL Server: (y/N)

- Whether to configure SSL settings for Postgres Server.
- OpenText Information Archive setup only provides basic configuration initialization. Manual review and additional configuration customization is likely necessary for most deployments.
- See official PostgreSQL documentation for more details

SSL server certificate file (`ssl_cert_file`) for PostgreSQL Server:

- The IA Server's public key certificate for clients to trust in order to establish an SSL connection.

SSL server private key file (`ssl_key_file`) for PostgreSQL Server:

- The IA Server's private key used to establish an SSL connection

SSL client certification for PostgreSQL Server {1=no, 2=verify-ca, 3=verify-full}: (3)

- Whether or not to enable client certificate authentication.
- Verify-CA merely verifies the certificate signer, Verify-FULL also verifies the client host to match the one in the client certificate and is thus both more strict and secure but also more expensive

SSL trusted certificate authorities file (`ssl_ca_file`) for client certificate validation for PostgreSQL Server:

- The CA certificate used to sign client certificates the PG Server should trust

SSL certificate revocation list file (`ssl_crl_file`) for blocked client certificates, optional leave blank for none for PostgreSQL Server:

- The file specifying the list of client certificates that should not be allowed despite being signed by a trusted CA

13.8.5 OTDS initialization configuration settings

OTDS Initialization Information Archive IAWA Redirect URL: This can be multiple values and must be comma separated. Each URL must also end with a `/`.

If you are planning to use external Apache Tomcat, it is recommended that you set two values. You can always review and correct the values from OTDS Admin when viewing the settings for the Oauth2 client for `infoarchive.iawa`. For example:

```
https://10.9.203.252:8080/, https://10.9.203.252:9443/infoarchive-webapp/
```

From OTDS Admin:

Redirect URLs

`https://<hostname>:8080/`
 	
`https://<hostname>:9443/infoarchive-webapp/`

In the example above, `9443` represents the SSL port for the external Tomcat and this cannot be the same Tomcat hosting OTDS.

OTDS Initialization Information Archive Resource secret key (16 characters):

Tip: Use an automatically generated password that is exactly 16 characters in length; otherwise, you will be prompted to reenter a 16 character password.

OTDS Initialization Information Archive Gateway client secret:

- Use an automatically generated password

OTDS Initialization Information Archive IAWA client secret:

- Use an automatically generated password

OTDS Initialization Information Archive CLI client secret:

- Use an automatically generated password

OTDS Initialization Information Archive JDBC client secret:

- Use an automatically generated password

OTDS Initialization Information Archive SAP Connector client secret:

- Use an automatically generated password

OTDS Initialization Information Archive Server client secret:

- Use an automatically generated password

Generate OTDS Initialization Information Archive Example partition configuration:

```
Generate OTDS Initialization Information Archive Example partition configuration: (y/N)
y

Password for example users:
XXXXXXXXXXX

Repeat Password for example users:
XXXXXXXXXXX
```

13.8.6 IA Server questions

Configure Information Archive Server: (Y/n)

- Whether or not to generate configuration files for IA Server

13.8.6.1 Crypto

Memory cache questions

Enable cross transactional (in memory) crypto key cache: (Y/n)

- see the preceding note logged by setup about whether or not you want the cache enabled
- Enabling the cache can significantly improve performance at the cost of reduced protection of encryption keys by caching them in memory for longer than strictly needed.

Max # of keys in cache: (100)

- The total number of keys in the cache. When new keys need to be cached while it is full, older keys are evicted based on the eviction strategy.

Cached key expiration (in seconds): (3600)

- Keys older than this age are eligible to be evicted from the cache periodically.

Cache refresh interval (in seconds): (300)

- Interval at which to cleanup the cache, evicting expired keys.

Cached Key Eviction Strategy {1=LRU, 2=LFU}: (2)

- Strategy to determine which keys to evict from the cache to make room for new keys when maximum number of not yet expired keys have been cached.
- Least Recently Used => Keys are removed that have not been used for the largest amount of time, regardless of how often they were used while in the cache.
- Least Frequently Used => Keys are removed that were least often used while in the cache, regardless of when they were last used.

Crypto keystore storage strategy {1=DATABASE, 2=KEY_MEDIATOR}: (1)

- For production systems DATABASE should be used. Key Mediator is currently an incubating feature for experimental use only.
- It is currently not possible to change this strategy once IA Server has been started successfully.

Enable use of Gemalto for data encryption: (y/N)

- Requires third–party Gemalto jars on the classpath of IA Server even when configuration file password encryption does not use Gemalto

Gemalto Safenet Client properties file:

- Path to Gemalto specific configuration file

Database keystore settings

Crypto keystore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (3)

- Type of keystore when persisting it in the database

Crypto keystore password:

- The password for the crypto keystore to protect against unauthorized key access

Key Mediator

Key Mediator Service Host:

Key Mediator Service Port: (8443)

Use SSL for Key Mediator: (Y/n)

Key Mediator Service Initial MEK alias:

- This value is only used the first time the server starts up. Changing it afterwards will not have any effect.
- If you want to change the existing mek alias after initial startup, see the IA Shell command `set-mek-alias`. MEK is short for master encryption key and this alias represents the id/name of that key used as part of the URL when asking Key Mediator to encrypt or decrypt keys.

Key Mediator Service Tenant Id:

- Optional value used for tracking calls to Key Mediator

Key Mediator authentication strategy {1=API_KEY, 2=OTDS}: (1)

- Whether to authenticate against KeyMediator using an API key or OTDS

Key Mediator Service API key:

- Refer to Key Mediator documentation on how to generate this, only required for API_KEY authentication strategy

Key Mediator with OTDS

OTDS host for Key Mediator:

OTDS Port for Key Mediator: (8443)

Use SSL for OTDS for Key Mediator: (Y/n)

- Refer to Key Mediator documentation, typically this would be true for production environments

Key Mediator OTDS Client Id:

- OTDS Client ID for the KeyMediator client integrated in OpenText Information Archive used for OTDS authentication

Key Mediator OTDS Client Secret:

- OTDS Client Secret for the KeyMediator client integrated in OpenText Information Archive used for OTDS authentication

13.8.6.2 System database

Host for system psql database cluster:

Port for system psql database cluster(<= 65536): (5432)

Username for system psql database cluster:

- The username for the data node owner. OpenText Information Archive uses this account to determine if a database exists, and creates databases if it has permission to do so.
- If this user account does not have database creation permission, databases need to be created in PG outside of IA Server beforehand instead.

Password for system psql database cluster:

- Password for the datanode owner.

Username for system psql database:

- Conceptually the username of the system data database owner.
- If the database is created beforehand, this username should match the database owner to avoid PG permission based problems.
- Can be the same user account as the datanode owner, but as a best practices it should be a different user account.

Password for system psql database:

- Password for the system data database owner. In case the datanode and database owner is the same account/username, this question will not be asked to avoid possible ambiguity in case of differently configured passwords.

13.8.6.2.1 SSL for system database

Use SSL for system psql database cluster: (y/N)

SSL PEM encoded X509v3 root certificate (sslrootcert) for server certificate validation for system psql database cluster:

SSL mode (sslmode) for system psql database cluster {1=disable, 2=allow, 3=prefer, 4=require, 5=verify-ca, 6=verify-full}: (6)

Enable 2-way SSL: (y/N)

- Whether to configure client certificate authentication for PG SSL connections for the system data database

SSL client private key (sslkey). PKCS-8 DER key (*.der) or PKCS-12 keystore (*.p12 or *.pfx) file for system psql database cluster:

- The private key file for the datanode owner account/IA Server to PG datanode connection
- Can be an individual private key file, or a keystore file containing the private key.

Is the SSL key for system psql database cluster password protected: (Y/n)

- Private keys can optionally be password protected, this question is asked to identify whether to force the password to be provided.

SSL key password (sslpassword) for system psql database cluster:

- Specify the private key password if the key is password protected as indicated by the previous question.

SSL PEM encoded X509v3 client certificate (sslcert) for system psql database cluster:

- The certificate file for datanode owner SSL connections

Use user/role specific SSL Client Certificate Authentication for system psql database: (Y/n)

- Client certificate authentication can either authenticate just the machine on SSL level, but it can also be used to authenticate the user account to use for the connection.
- This would require a separate private key and corresponding client public key certificate per user account.
  - In case of different user accounts for data node and databases, this needs a separate key+certificate for datanode and database connections
  - In case of a dedicated user account per database, this would even need a separate key+certificate per database

SSL client private key (`sslkey`). PKCS-8 DER key (*.der) or PKCS-12 keystore (*.p12 or *.pfx) file for system psql database:

- private key for database SSL connections

SSL PEM encoded X509v3 client certificate (sslcert) for system psql database:

- public key certificate for database SSL connections

13.8.6.3 General server settings

Server Listen Address: (0.0.0.0)

- The IP address IA Server needs to listen for/accept connections on. This primarily affects multi-home machines and affects through which networks IA Server can be connected to.
- A listen address of 127.x.x.x for example would only allow connections from the same machine.
- A listen address of 0.0.0.0 on the other hand would listen for connections on all IP's of the machine.

Server Port: (8765)

Server loglevel threshold {1=DEBUG, 2=INFO, 3=WARN, 4=ERROR}: (3)

- Represents the IA Server global log–level threshold. Anything less relevant than the threshold will not be logged unless explicitly configured otherwise.
- Does not affect the archived diagnostics logs for most background processing (jobs/order items). These are controlled by the loglevel on the job or corresponding programmatic overrides.

SSL for server

Enable Server SSL (HTTPS): (y/N)

Server SSL keystore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (1)

Server SSL keystore:

Server SSL keystore password:

Server SSL key alias:

Server SSL key password:

Server SSL truststore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (2)

Server SSL truststore:

Server SSL truststore password:

Server SSL client authentication {1=NONE, 2=WANT, 3=NEED}: (3)

13.8.6.4 Optional profiles

Enable OTDS: (y/N)

- Enabling OTDS requires additional manual configuration.

Disable all background processing: (y/N)

- By default each IA Server node performs various kinds of background processing. Each type can be disabled individually by means of a profile, but as a convenience there is a grouped profile to disable all background processing on specific IA Server nodes.
- It is important that there are enough IA Servers that do perform background processing and at least one at all times.
- The exception is when disaster recovery is being performed, in which case all background processing needs to be disabled.

Enable OTIV: (y/N)

- Whether to enable Server-side integration of OT Intelligent Viewing

Enable Metrics: (y/N)

- Whether to enable system process metrics (primarily for cloud deployments)

Kafka questions

Enable Kafka: (y/N)

- Whether to enable Kafka integration. This requires additional manual configuration

Kafka bootstrap servers:

- This is specific to Kafka configuration

Kafka default topic: (com.opentext.infoarchive)

Kafka client id: (infoarchive-server-1)

Kafka security protocol {1=PLAINTEXT, 2=SASL_SSL, 3=SASL_PLAINTEXT, 4=SSL}: (1)

Kafka SASL mechanism {1=PLAIN, 2=SCRAM-SHA-256, 3=SCRAM-SHA-512, 4=AWS_MSK_IAM}: (1)

Kafka SASL jaas config:

- Consult Kafka documentation on how to set this value. There cannot be any newlines in the value and this value is considered sensitive information so the user cannot see what they are typing.

Kafka SASL client callback handler class:

13.8.7 IA Gateway questions

Gateway Server (connect to Gateway) Listen Address: (0.0.0.0)

Gateway Server (connect to Gateway) Port: (8080)

Gateway loglevel threshold {1=DEBUG, 2=INFO, 3=WARN, 4=ERROR}: (3)

Enable OTDS SSO authentication: (y/N)

- Requires additional manual steps

Information Archive gateway token secret:

13.8.7.1 Gateway SSL questions

Enable Gateway Server (connect to Gateway) SSL (HTTPS): (y/N)

Gateway Server (connect to Gateway) SSL keystore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (1)

Gateway Server (connect to Gateway) SSL keystore:

Gateway Server (connect to Gateway) SSL keystore password

Gateway Server (connect to Gateway) SSL key alias:

Gateway Server (connect to Gateway) SSL key password:

Gateway Server (connect to Gateway) SSL truststore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (2)

Gateway Server (connect to Gateway) SSL truststore:

Gateway Server (connect to Gateway) SSL truststore password:

Gateway Server (connect to Gateway) SSL client authentication {1=NONE, 2=WANT, 3=NEED}: (3)

13.8.7.2 Server connection questions

Gateway Client (connect to Information Archive Server) Host:

Gateway Client (connect to Information Archive Server) Port: (8765)

13.8.7.3 Server connection SSL questions

Enable Gateway Client (connect to Information Archive Server) SSL (HTTPS): (y/N)

Gateway Client (connect to Information Archive Server) SSL keystore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (1)

Gateway Client (connect to Information Archive Server) SSL keystore:

Gateway Client (connect to Information Archive Server) SSL keystore password:

Gateway Client (connect to Information Archive Server) SSL key alias:

Gateway Client (connect to Information Archive Server) SSL key password:

Gateway Client (connect to Information Archive Server) SSL truststore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (2)

Gateway Client (connect to Information Archive Server) SSL truststore:

Gateway Client (connect to Information Archive Server) SSL truststore password:

13.8.7.4 Optional profiles

Enable Smart URL: (y/N)

Enable CORS: (y/N)

Enable OTIV: (y/N)

13.8.8 IA Shell questions

Configure Information Archive Shell: (Y/n)

13.8.8.1 Connection questions

Information Archive Server (for Shell) Host:

Information Archive Server (for Shell) Port: (8765)

Gateway (for Shell) Host:

Gateway (for Shell) Port: (8080)

Gateway Cli (IA Shell) client secret:

13.8.8.2 IA Server SSL

Enable Information Archive Server (for Shell) SSL (HTTPS): (y/N)

Information Archive Server (for Shell) SSL keystore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (1)

Information Archive Server (for Shell) SSL keystore:

Information Archive Server (for Shell) SSL keystore password:

Information Archive Server (for Shell) SSL key alias:

Information Archive Server (for Shell) SSL key password:

Information Archive Server (for Shell) SSL truststore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (2)

Information Archive Server (for Shell) SSL truststore:

Information Archive Server (for Shell) SSL truststore password:

13.8.8.3 Gateway SSL

Enable Gateway (for Shell) SSL (HTTPS): (y/N)

Gateway (for Shell) SSL keystore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (1)

Gateway (for Shell) SSL keystore:

Gateway (for Shell) SSL keystore password:

Gateway (for Shell) SSL key alias:

Gateway (for Shell) SSL key password:

Gateway (for Shell) SSL truststore type {1=PKCS12, 2=JKS, 3=JCEKS, 4=BKS}: (2)

Gateway (for Shell) SSL truststore:

Gateway (for Shell) SSL truststore password:

13.8.9 Application questions

Configure Applications: (Y/n)

- To install table–based applications the application configuration needs to identify PG datanode and PostgreSQL database connection information for which setup can generate IA Shell default properties.
- These questions only populate 'default' properties which may or may not actually be used by individual application configuration files.
- This is mostly intended as a convenience feature that may not be practical for production deployments where these values might necessarily differ between applications, which setup cannot accommodate for.

13.8.9.1 Default structured data connection

Host for Default PostgreSQL structured data database cluster:

- IP address/hostname for the structured data PostgreSQL datanode/database cluster, which may be the same or a different datanode as used for system data.
- For production it is recommended to use a separate datanode for both, or even more than one for structured data depending on data volume, but this is not strictly required and a single datanode can be shared if needed.

Port for Default psql structured data database cluster(<= 65536): (5432)

Username for Default psql structured data database cluster:

- The username for the data node owner. OpenText Information Archive uses this account to determine if a database exists, and creates databases if it has permission to do so.

Password for Default psql structured data database cluster:

Username for Default psql structured data database:

Password for Default psql structured data database:

13.8.9.2 Default structured data SSL connection

Use SSL for default psql structured data database cluster: (Y/n)

SSL PEM encoded X509v3 root certificate (`sslrootcert`) for server certificate validation for Default psql structured data database cluster:

SSL mode (sslmode) for Default psql structured data database cluster {1=disable, 2=allow, 3=prefer, 4=require, 5=verify-ca, 6=verify-full}: (6)

Enable 2-way SSL: (y/N)

SSL client private key (sslkey). PKCS-8 DER key (*.der) or PKCS-12 keystore (*.p12 or *.pfx) file for Default psql structured data database cluster:

SSL PEM encoded X509v3 client certificate (sslcert) for Default psql structured data database cluster:

Use user/role specific SSL Client Certificate Authentication for Default psql structured data database: (Y/n)

SSL client private key (sslkey). PKCS-8 DER key (*.der) or PKCS-12 keystore (*.p12 or *.pfx) file for Default psql structured data database:

SSL PEM encoded X509v3 client certificate (`sslcert`) for Default psql structured data database:

13.8.10 SAP Connector

Configure SAP Connector: (y/N)

- Whether to generate SAP Connector configuration files
- Optional and only applicable when using SAP Connector

Information Archive authentication Gateway Host (for SAP Connector):

- IP/hostname of the IA Web App/Gateway SAP Connector should use for authenticate purposes

IA authentication Gateway Port (for SAP Connector): (8080)

IA Server Host (for SAP Connector):

- IP/hostname of the IA Server for SAP Connector to connect to

IA Server Port (for SAP Connector): (8765)

SAP Connector client secret:

SAP Connector authentication user:

- name of user that can connect to OpenText Information Archive (see SAP Connector documentation for more details)

SAP Connector authentication password:

- password of the user to connect to OpenText Information Archive (see SAP Connector documentation for more details)

13.8.11 Upgrade questions

Upgrade from existing Information Archive installation? (Y/n)

- This will make setup inherit configuration settings from an existing installation and skip questions for components that were not configured in the existing installation.
- Can be used for upgrading from the last previous version or when deploying a patch into a new installation location.

Location of existing Information Archive installation:

- Specify the root installation folder of the previous IA installation to inherit configuration files from.

13.9 Deploying IA Web App to Apache® Tomcat™

Instead of installing IA Web App as a standalone Spring Boot application with an embedded Apache Tomcat server, you can deploy IA Web App as a WAR web application in an external Apache Tomcat server. For the supported versions of Apache Tomcat, see the *OpenText Information Archive Release Notes*.

Important

If you updated the 25.2 version of the `spring.security.oauth2.client.registration.infoarchive.iawa` section in the `<IA_ROOT>/config/iawebapp/application-infoarchive.gateway.profile.OTDS.yml` file, you must manually update the section again for the 25.3 version. During development of the 25.3 version of the product, the section was renamed to `spring.security.oauth2.client.registration.infoarchiveIawa`. Consequently, during an upgrade, this particular section does not "inherit" any values that were manually entered in 25.2.

Note: If you deploy IA Web App to Apache Tomcat and use OTDS in SSO mode, using the `infoarchive.gateway.profile.OTDS` profile, then there is a race condition with the startup of IA Web App and OTDS. This is because IA Web App expects that OTDS is already up and serving on the certificate endpoint. IA Web App uses the OTDS certificate to verify the signature of JWT tokens issued by OTDS. The order in which IA Web App and OTDS start is not predictable. Therefore, you should deploy IA Web App and OTDS to separate instances of Apache Tomcat. Also, you should make sure that OTDS is up and running before you start IA Web App.

By default, the logs are in the `<TOMCAT_ROOT>/bin/logs/iawebapp` directory. You can change the log file location.

Before you begin:

- Extract the distribution (`infoarchive.zip`) file to the `<IA_ROOT>` directory. Copy the *<<IA_ROOT>/lib/infoarchive-webapp.jar>* onto the machine hosting the external Tomcat.

- Install Apache Tomcat. Keep track of where it was installed. This guide refers to this location as `<TOMCAT_ROOT>`. The redirect URL for the IA Web App OAuth2 client needs to include the new URL for the external Tomcat. This can be viewed and changed using OTDS Admin.
- Configure Tomcat to enable SSL, which can be done by configuring the `server.xml` file in the Tomcat `config` directory. Refer to the Apache Tomcat documentation for further instructions.
- Verify that you have already installed IA Server on a separate computer and that the IA Server is running.
- Generate the configuration for IA Web App.

To deploy IA Web App to Apache Tomcat:

1. Stop the Tomcat server.
2. Copy the `infoarchive-webapp.jar` file from the temporary location to the Tomcat server's `webapps` directory (for example, `<TOMCAT_ROOT>/webapps`).
3. Rename the `infoarchive-webapp.jar` file to `infoarchive-webapp.war`.
4. Start the Tomcat server.

   The Tomcat server expands the `infoarchive-webapp.war` file into the `<TOMCAT_ROOT>/webapps/infoarchive-webapp` directory.
5. Stop the Tomcat server.
6. Copy the following configuration files for IA Web App to the `<TOMCAT_ROOT>/webapps/infoarchive-webapp/WEB-INF/classes` directory:

   - `<IA_ROOT>/config/iawebapp/application.yml`
   - `<IA_ROOT>/config/iawebapp/application-CLIENTS.yml`

   You can also copy other profiles and configuration files, depending on your requirements for an OpenText Information Archive deployment (for example, configuration files for OTDS). If using Vault, be sure to copy the `application-vault.properties` file, too.

   Note: If, at any time, you change the configuration files in the `<TOMCAT_ROOT>/webapps/infoarchive-webapp/WEB-INF/classes` directory, you must restart the Tomcat server.
7. In Tomcat, change the `cacheMaxSize` to `51200`, as this is required for the OpenText™ Brava!™ viewer components embedded in the `infoarchive-webapp.war` file. Change the context XML in the `<TOMCAT_ROOT>/config` to add an extra line:

```
<Context>

    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
```

```
    <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

    <Resources cacheMaxSize="51200" />
    ...
</context>
```

8. In the `application.yml` file:

   a. Ensure the correct port for the external Tomcat is listed (setting is `infoarchive.gateway.port`). Also, ensure the value for the SSL port is correct, as Tomcat supports both `http` and `https` protocols.

   b. Set the IA Web App's context path with the `infoarchive.gateway.contextPath` property. The value should be `/infoarchive-webapp/` after you renamed the `WAR` file in the previous step.

9. Optional It is strongly recommended that you set up and correctly configure an `application-csp.yml` file. It is not included in the WAR file, but setup does create it. If the file is not configured properly, it could cause issues with login for retrieving the OAuth2.0 token. This file is should be stored in the `<TOMCAT_ROOT>/webapps/infoarchive-webapp/WEB-INF/classes` directory.

10. Optional To configure the location for the internal `webapp` logs, complete one of the following:

    - Update the default logs directory (`LOGS_DIR`) in the `<TOMCAT_ROOT>/webapps/infoarchive-webapp/WEB-INF/classes/logback-spring.xml` file. For example, change the following line:

      ```
      <property name="LOGS_DIR" value="${logsdir:-logs/iawebapp}" />
      ```

      To this:

      ```
      <property name="LOGS_DIR" value="${logsdir:-c:/logs/iawebapp}" />
      ```

    - Use `APACHE_TOMCAT` special attributes and properties by setting up `LOGS_DIR`.

11. Restart the Tomcat server.

12. To verify that IA Web App is running, open a supported browser, and go to `<IP_OF_IAWEBAPP>:<PORT_NUMBER>/infoarchive-webapp`. If the login screen appears, then IA Web App is running. Login to the system with valid credentials.

13. If you are no longer using the standalone version of the IA Web App, modify the IA Shell configuration to update the `gatewayURL`.

13.10 How example applications demonstrate product features

Every example application in the OpenText Information Archive distribution demonstrates specific features of the product. The following tables include a list of feature descriptions that each application can demonstrate.

Important
Some sample applications have a `README` file that provides instructions on how to install and use the application.

13.10.1 Declarative configuration and data structures

The following table outlines the declarative configuration (DC) structure for each example IA application, along with the data structure and encryption features for each:

Feature	Description
DC structure	Indicates the way the application sources are presented. It is possible to have one of the following: • *A role-based structure* where YML files are divided into modules: the administration module, application module, and compliance module • *A nested structure*, including external YML files. For more information about the declarative configuration, see section 8 "Declarative configuration" in *OpenText Information Archive - Configuration Guide (EARCORE-CGD)*.
Data structures	• *Application Type*: Indicates if the application is table- or SIP-based. For more information, see section 2.3 "Table archiving and SIP archiving" in *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*. • *Data Types*: Indicates whether the application supports either structured data only or structured and unstructured data together. • *Other*: Indicates any other unclassified demonstration feature.

Application	DC Structure	Data Structures		
		Application Type	Data Types	Other
Baseball	Role-based	Table	Structured and unstructured	Two schemas

Application	DC Structure	Data Structures		
		Application Type	Data Types	Other
Certificates	Role-based	SIP	Structured and unstructured	
CertificatesAndTrades	Role-based	SIP	Structured and unstructured	Two holdings
CreditTresor	Role-based	SIP	Structured and unstructured	Setting permissions and retention policies
Invoices	Role-based	SIP	Structured and unstructured	PDF file processing
OrderManagement	Role-based	Table	Structured data	Database constraints
Patent	Role-based	Table	Structured data	
PhoneCalls	Role-based	SIP	Structured and unstructured	
PhoneCallsGranular	Application structure based on modules without paying attention to roles	SIP	Structured and unstructured	Apply retention and holds using rules. Event-based retention.
SAPConnector	SIP. The application configuration for SAP archiving. For more information, see section 1.7 “Configuration and administration” in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.			
SEPA	Role-based	SIP	Structured data	
SharePoint	SIP. The application configuration for archiving data extracted from SharePoint. For more information, see section 2.2 “Configuring and running SharePoint connector” in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.			
Tickets	Role-based	SIP	Structured and unstructured	

Application	DC Structure	Data Structures		
		Application Type	Data Types	Other
Trades	Role-based	SIP	Structured and unstructured	

13.10.2 Encryption for ingestion and search

Feature	Description
Encryption	Indicates whether the application has: • Encryption for data preservation in a database • Decryption capabilities to search the encrypted data For more information, see the *OpenText Information Archive - Encryption Guide (EARCORE-AGE)*.

Application	Encryption
Baseball	Yes
Certificates	Yes
CertificatesAndTrades	Yes
CreditTresor	Yes
Invoices	Yes
OrderManagement	No
Patent	No
PhoneCalls	Yes
PhoneCallsGranular	Yes
SAPConnector	The application configuration for SAP archiving. For more information, see section 1.7 “Configuration and administration” in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.
SEPA	No
SharePoint	The application configuration for archiving data extracted from SharePoint. For more information, see section 2.2 “Configuring and running SharePoint connector” in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.
Tickets	Yes

Application	Encryption
Trades	Yes

13.10.3 Search features

Feature	Description
Nested Search	Indicates if the application includes a demo configuration for nested searches. For more information, see section 6.8.2 “Nested searches” in *OpenText Information Archive - Configuration Guide (EARCORE-CGD).*
Custom Presentation	Indicates if the application demonstrates the custom search presentation capabilities.
Value List	Indicates if the application demonstrates the value list capabilities for the search. For more information, see section 6.5.13 “Value lists” in *OpenText Information Archive - Configuration Guide (EARCORE-CGD).*
Other	Indicates any other unclassified demonstration feature.

Application	Nested Search	Custom Presentation	Value List	Other
Baseball	No	Yes	Yes	• Result export • Custom search form • Data resolution (with extended search) • XQuery drop-down • HTML content view column • Encrypted field search • Full-text search • Dynamic column
Certificates	No	No	No	
CertificatesAnd Trades	Yes	No	Yes	• Cross-holding search • XQuery drop-down

Application	Nested Search	Custom Presentation	Value List	Other
CreditTresor	Yes	No	Yes	• Result export
CrossApplicationSearch	No	No	No	• Cross-application search*
Invoices	No	Yes	No	• Result export • PDF viewer • Extended search
OrderManagement	Yes	Yes	Yes	• Dependency drop-down
Patent	No	No	No	• Full-text search
PhoneCalls	Yes	Yes	No	• Result export • Custom search form (with context information) • Custom presentation (with context information)
PhoneCalls Granular	No	Yes	No	• Result export
SAPConnector	The application configuration for SAP archiving. For more information, see section 1.7 “Configuration and administration” in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.			
SEPA	No	No	No	• Result export
SharePoint	The application configuration for archiving data extracted from SharePoint. For more information, see section 2.2 “Configuring and running SharePoint connector” in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.			
Tickets	No	Yes	No	• Image viewer
Trades	No	No	Yes	• Data resolution • XQuery drop-down

Note: *Installing CrossApplicationSearch is different from the other sample applications in that it does not create a new application but contains a cross-application search that is anchored to the Certificates application and adds a

delegate search in both the Certificates application and the Trades application. Both Certificates and Trades must be installed before installing CrossApplicationSearch. In addition, CrossApplicationSearch defines the mapping required between the parent cross-application searches and the delegate searches.

13.10.4 Compliance features

Feature	Description
Retention	Indicates the type of retention policy included in the application configuration. For more information, see section 9.4 "What is a retention policy?" in *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*.
Hold	Indicates the type of hold included in the application configuration. For more information, see section 9.5 "What is a hold?" in *OpenText Information Archive - Fundamentals Guide (EARCORE-ACS)*.
Rule	Indicates if the application configuration includes a rule specification. For more information, see section 9.3.7.1 "Viewing an application's rules" in *OpenText Information Archive - Configuration Guide (EARCORE-CGD)*.

Application	Retention	Hold	Rule
Baseball	None	None	No
Certificates	Duration	Legal	No
CertificatesAndTrades	Duration	Legal	No
CreditTresor	Duration	Legal	No
Invoices	Duration	Legal	No
OrderManagement	None	None	No
Patent	Duration	Legal	Yes
PhoneCalls	Duration	Legal	No
PhoneCallsGranular	Mixed and event policies	Legal	Yes
SAPConnector	The application configuration for SAP archiving. For more information, see section 1.7 "Configuration and administration" in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.		
SEPA	None	None	No
SharePoint	The application configuration for archiving data extracted from SharePoint. For more information, see section 2.2 "Configuring and running SharePoint connector" in *OpenText Information Archive - SAP and SharePoint Connectors Guide (EARCORE-AGC)*.		

Application	Retention	Hold	Rule
Tickets	None	None	No
Trades	Duration	Legal	No

13.10.5 Permissions features

The `restrictRolesToApplicationGroups` feature specifies that application permissions also take into account role mapping.

For customers upgrading from a previous version of OpenText Information Archive, the `restrictRolesToApplicationGroups` feature is disabled. By default, the feature is enabled for new customers.

The CreditTresor application demonstrates how this feature can be used. A `README` file located in the `<IA_ROOT>\examples\applications\CreditTresor` directory provides more information about using declarative configuration (DC), and information about configuring applications and retention policies to have permissions.

Also, navigate to the **Permissions** tab. As an Administrator, ensure the following:

- The **Category** field is set to **Application**.
- The **Application** field is set to **CreditTresor**.

By default, only the `GROUP_ADMINISTRATOR`, `GROUP_BUSINESS_OWNER`, and `GROUP_END_USER` can access the application.

As a consequence, if `restrictRolesToApplicationGroups` is enabled, when you log in as Sue (who is in the Retention Manager and Developer groups), you still cannot perform Retention Manager actions, such as applying holds to search results, or Developer actions, such as editing a search definition.

13.10.6 Unstructured content extraction feature

OpenText Information Archive supports the extraction, storage, indexing and querying of text from unstructured content, both for SIP and table applications. Although further configuration work must be done, the feature is off, by default. This enables you to opt in through the `ingestion.ci.text.enable` parameter in the **Global Settings** tab. For more information, see section 3.12 “Global Settings” in *OpenText Information Archive - Administration Guide (EARCORE-AGD)*.

The OrderManagement table application showcases the feature. It allows you to search through fact sheets in the PARTS table.

For SIP, check out the Invoices application, which lets you search through unstructured content attached to invoices.

13.11 Declarative configuration basics for OpenText Information Archive applications

Note: Some of the code samples shown in this guide contain `\` newline characters that allow the code to flow over several lines for readability and formatting purposes. The `\` newline characters are not to be used as part of the code shown, and you must manually remove them prior to using the code.

When you run the `install.bat` (Windows) or `install` (Linux) script for a declarative configuration IA application, you are effectively running the IA Shell commands that are specified in the IA application's `install.iashell` file. You can edit this file to run IA Shell commands, and you can comment out unwanted lines by using two slash symbols (`//`).

For example, for the Baseball application, you can edit the default `install.iashell` file to comment out the `ingest-tables` and `index-build` commands and run a `chain-of-custody` command:

```
connect
import config
// ingest \
   --d applications/Baseball/databases/Baseball-sql-db/schemas/BASEBALL \
   --path data/BASEBALL
// ingest \
   --d applications/Baseball/databases/Baseball-sql-db/schemas/TICKETS \
   --path data/TICKETS
// index-build \
   --d applications/Baseball/databases/Baseball-sql-db \
   chain-of-custody --d applications/Baseball/databases/Baseball-sql-db/
schemas/BASEBALL \
   --file ../../../config/iashell/chain-of-custody-schema.xml
quit
```

Note: With declarative configuration IA applications, OpenText Information Archive applies all the actions that it can for a transaction and generates error messages for those actions that it cannot apply (for example, `Skipped xquery: Record Id Search (not changed)`). If you run an `install.bat` or `install` script with several commands, each command is a separate transaction.

If you do not want to ingest any data on installation, edit the IA application's `install.iashell` file and remove the `ingest` command.

Instead of using an `install.bat` or `install` script, you can start an IA Shell session and enter any commands manually. The IA Shell commands that can perform equivalent functionality to the Ant targets are as follows:

- `import`
- `receive`
- `ingest`
- `chain-of-custody`

- `view-chain-of-custody`

For more information about IA Shell commands, see section 2 “IA Shell commands” in *OpenText Information Archive - IA Shell Guide (EARCORE-ARE)*.