Windows Install Guide

OpenText™ Intelligent Viewing

Install and configure Intelligent Viewing on Windows.

CLIVSA250400-IGD-EN-01

Windows Install Guide

OpenText™ Intelligent Viewing

CLIVSA250400-IGD-EN-01
Rev.: 2025-Oct-02

This documentation has been created for OpenText™ Intelligent Viewing 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

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Table of Contents

Part 1

Introduction

Chapter 1

Product overview

OpenText Intelligent Viewing is a cloud and off cloud product built for file viewing and transformation. Supported features include annotations, redactions, text and graphical comparisons of documents and drawings, measurement of drawings, publishing to PDF or TIFF, and the ability to combine documents in a single view and then publish them as a single document.

For a full list of the document formats supported by Intelligent Viewing, see supported document types (https://www.opentext.com/assets/documents/en-US/pdf/opentext-intelligent-viewing-supported-formats-en.pdf).

The product consists of seven services that can be integrated into an application: Asset Service, Configuration Service, Markup Service, Publication Service, Publisher Service, Highlight Service, and Viewer Service. **A functional IV installation requires at least one instance of each of these services, but some services may benefit from having multiple instances installed (on separate nodes).**

In addition to the seven core services, an installation requires three supporting services: a running RabbitMQ service, a running OpenText Directory Service (OTDS) and access to a database. These can be dedicated to IV or shared with other resources. Typically, each would exist on their own node but could also be co-hosted with some or all of the other services if the expected load is low.

1.1 Audience

This Installation Guide is written for system administrators or users with administrative privileges who are responsible for installing Intelligent Viewing. A base knowledge of database configuration, OpenText Directory Services (OTDS), and message queueing services is strongly recommended.

1.2 IV service overview

This seven services are grouped by functionality into the Transformation services and the Viewing services and are explained in the following sections.

1.2.1 Services architectural diagram

Intelligent Viewing consists of Transformation (publishing) and Viewing capabilities, which are delivered by several cooperating microservices as shown in the architecture diagram. It also depends on external services including the OpenText Directory Service (OTDS), a Database (PostgreSQL, Oracle, or Microsoft SQL Server), and a message broker (RabbitMQ) for proper operation.

Note: The components shown in green use Hazelcast clustering to communicate asynchronously using an event bus to share the data between services.

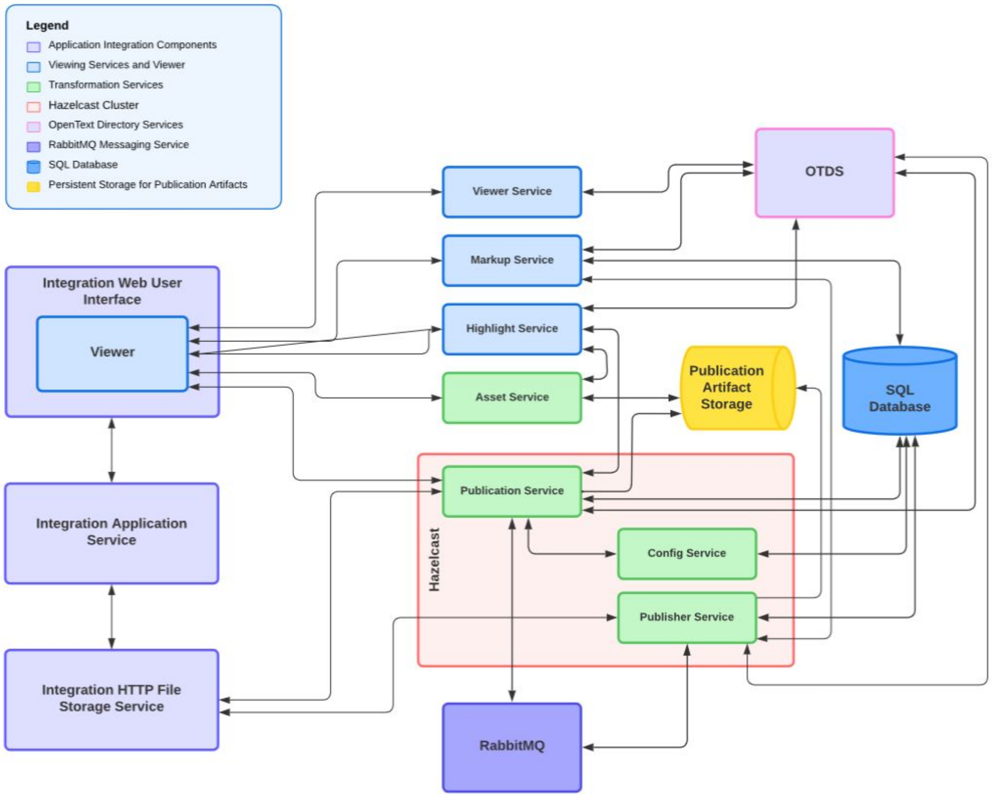

Legend
Application Integration Components
Viewing Services and Viewer
Transformation Services
Hazelcast Cluster
OpenText Directory Services
RabbitMQ Messaging Service
SQL Database
Persistent Storage for Publication Artifacts
OTDS
Viewer Service
Markup Service
Highlight Service
Asset Service
Integration Web User Interface
Viewer
Publication Artifact Storage
SQL Database
Hazelcast
Publication Service
Config Service
Publisher Service
Integration Application Service
Integration HTTP File Storage Service
RabbitMQ

Figure 1-1: Services architectural diagram

1.2.2 Transformation services

Transformation services handle the conversion of source documents into renditions that consist of rendered artifacts.

- **Publication Service:** This is the primary service that controls the transformation process and provides a REST API for initiating requests to perform document transformations. Publications are created, queried for status, and retrieved upon completion using the Publication Service: Creating a new publication starts the transformation by first using the Configuration Service to create a valid collection of settings for a publishing request and then placing the resulting request on a queue served by single/multiple Publishing Agents. The agent that processes the publishing request communicates its status in real time back to the Publication Service. It includes links to the artifacts it produces, which the service incorporates in the data for the newly created publication resource.

  As a creator and consumer of document transformations, the Publication Service is the only service that requires direct interaction. It is through this service alone that new publications are created, status is learned, and the location of rendered artifacts is discovered.

- **Configuration Service:** This service provides a REST API for discovering information about the options (called features) that are available when creating publications. It supports publication policies that can describe a particular type of transformation and the features that are used by that type, creating valid publishing requests based on these policies. The Configuration Service is used directly by the Publication Service for every publication it creates, but is not typically used directly by integrators.

- **Publisher Service:** This service is responsible for the actual transformation work of converting source documents into new renditions and publication artifacts (such as thumbnails and search text). These artifacts are stored by the Asset Service and the resulting addresses are stored into the publication data for each publication. Each Publisher Service instance is independent and works directly with the Publication Service. Users never directly interact with the Publisher Service.

  The Publisher can load a very large variety of source document formats into an OpenText proprietary intermediate format (XDL) used as the common basis for conversion to the supported publishing formats: PDF, TIFF, JPEG, PNG, text, SVG, and the proprietary XDL. The Publisher stores the artifacts it produces based on a publishing target, specified in the publication, which is typically a blob storage service or a shared persistent volume or file system accessible to the Publisher in the target environment. For example, in the OpenText SaaS environment, the OpenText Content Storage Service provides the blob storage, thus acting as the publishing target for artifact storage. As artifacts are generated, the Publisher reports the URLs of the stored artifacts and the publishing status, in real time, to the Publication Service. The Publisher creates threads for the following document types:

  – **DRW** (image formats)

  - **PDF** (PDF documents)
  - **DOC** (Microsoft Office and other text formats)

  There is no mechanism to interact directly with a Publishing Agent; it is only by using the Publication Service to create a new publication resource that a Publishing Agent can be tasked with doing any work.

- **Asset Service:** The Asset Service provides HTTP access to the publication artifacts, published by the Publishing Agent, to a volume storage shared by the Asset Service, Publishing Agent, and Publication Service. It uses the Publication Service's per-publication access control policy to provide access control for artifacts produced for each publication. The Asset Service does not provide its own REST API.

1.2.3 Viewing services

The viewing capability is provided by three cooperating microservices: the Viewer Service, the Highlight Service, and the Markup Service. These services are used to display and interact with source documents that have been transformed into a suitable publication. While all three microservices present APIs that allow direct interaction, typically the web viewer does most of the REST API calls on behalf of the user. The web viewer refers to the Integration Web User Interface.

- **Viewer Service:** The Viewer Service is responsible for providing a REST API for discovering and downloading an available web viewer, which interacts with the Highlight Service, Markup Service, Asset Service, and Publication Service from the client machine. This service has no dependency on any other microservice and can expose REST APIs for multiple different kind of web viewers.
- **Highlight Service:** The Highlight Service (formerly Search Service) is responsible for providing a REST API to search for text content in a given transformation, and return term hit highlight polygons or the text within a given polygonal region of a page. The Highlight Service relies upon the Publication Service to discover the requested transformation's text data. The service always verifies the requestor's authorization to access the transformation contents prior to returning any results.
- **Markup Service:** The Markup Service provides a GraphQL API for loading and storing annotations (sometimes called markups) that are made against a transformation. The annotations are associated with the tenant subscriptions found in the requests made to the service. The web viewer can retrieve annotations for a document being viewed using the Markup Service, can edit annotations, can author new annotations, and can store them into the database using the Markup Service.

1.2.4 Supporting services

Intelligent Viewing uses the following third-party services to provide solid and stable implementations of commonly available back-end services.

> ! **Important**
>
> These three services must be correctly installed before Intelligent Viewing is installed. The IV installer will look for them and, in some cases, apply the required configuration.

- **RabbitMQ:** This open-source message-broker is used to pass messages between the Publication Service and the Publisher Service(s) to track the status of transformation jobs. Only these two services need access to the RabbitMQ server.
- **OpenText Directory Services (OTDS):** This directory infrastructure product handles authentication and licensing for Intelligent Viewing.
- **Database:** Database tables are used by Intelligent Viewing to store several different types of information. Annotations are stored in a database table, associated with source documents. These are accessed through the Markup Service. Publications are also stored in a database table. The information in a publication is updated through the Publication Service. IV currently supports Microsoft SQL Server, PostgreSQL and Oracle databases.

1.3 Software platform

- **Node.js along with npm**: npm is a package manager for `Node.js`. The viewing services are written in JavaScript and run on `Node.js`, a cross-platform back-end JavaScript runtime environment. This environment must be installed before the viewing services are installed.
- **Java:** The transformation services are written in Java and JavaScript and run on the OpenJDK Java runtime environment using GraalVM for the JavaScript portions. The OpenJDK environment must be installed before the transformation services are installed. GraalVM is built into the services.

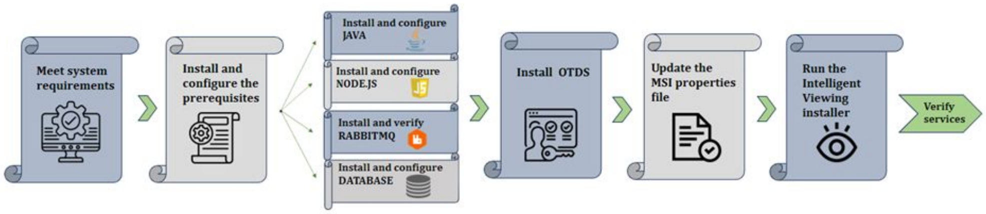
Part 2

Install and configure

Chapter 2

Installation overview

The flow chart below illustrates the basic steps required for a successful Intelligent Viewing installation.


Meet system requirements
Install and configure the prerequisites
Install and configure JAVA
Install and configure NODE.JS
Install and verify RABBITMQ
Install and configure DATABASE
Install OTDS
Update the MSI properties file
Run the Intelligent Viewing installer
Verify services

Requirements > PREREQ > JAVA > NODE.JS > RABBITMQ > DATABASE > OTDS > MSI > IV > Verify

For system requirements information, refer to the Intelligent Viewing Product Release Notes on My Support (https://support.opentext.com/).

Chapter 3

Environment setup

Before you can use Intelligent Viewing, you must prepare your environment with the necessary installation dependencies and services.

Dependencies can be installed by whatever method your organization prefers.

Service startup notes

- RabbitMQ, OTDS licensing, and your database (PostgreSQL, Microsoft SQL Server, or Oracle) must be started to run the services. If these applications are not started, the Intelligent Viewing services will not start.
- The order that services are started/stopped does not matter, but Windows Services might enforce dependencies when performing start and stop.
- If the Configuration Service is manually stopped, the Asset, Publication, and Publisher Services will also be stopped.
- If the Publication or Publisher Services are manually started individually, the Config Service will also start, with a slight delay.
- Performing an uninstall while the `services.msc` window is open can cause services to not be fully removed until the window is closed. This can potentially cause failure during reinstall.

3.1 Transformation services

Each machine that will be running one of the transformation services must have Java installed.

During the installation of the Publisher Service, the Chrome web browser and the LibreOffice office suite will be installed if they are not already installed. These applications are used by the Publisher Service for the conversion of certain file types.

Important

If LibreOffice is installed during the Intelligent Viewing installation, a message displays requiring a system restart. After restarting, the Intelligent Viewing installer must be relaunched to continue and complete the installation process. To avoid this interruption, an Administrator can download the latest stable version of LibreOffice from https://www.libreoffice.org/download/download-libreoffice/, install LibreOffice, then install Intelligent Viewing.

Install OpenJDK (version 17)

If Java is already installed, make sure the JAVA_HOME environment variable exists.

3.2 Viewing services

Each machine that will be running one of the viewing services must have `Node.js` and `npm` installed.

Install Node.js

To download `Node.js`, refer to https://nodejs.org/en/download. Include `npm` during the `Node.js` install (which is typically included by default).

Refer to the Intelligent Viewing product release notes on My Support (https://support.opentext.com/) to verify the correct version.

3.3 Supporting third-party services

Ideally, the supporting third-party services should each be installed on separate machines in a production environment. For demonstration purposes, they can all be included on the same machine.

3.3.1 Install RabbitMQ

To install RabbitMQ:

1. Install Erlang: RabbitMQ has its own pre-requisite, called Erlang, which needs to be installed before installing RabbitMQ. The supported Erlang version is dependent on the RabbitMQ version and a compatible version of Erlang must be installed. For details, see https://www.rabbitmq.com/which-erlang.html or https://erlang.org/download/.

2. Download the supported RabbitMQ version from https://www.rabbitmq.com/download.html. Refer to the Intelligent Viewing product release notes on My Support (https://support.opentext.com/) to verify the correct version.

   Install the software by following the product instructions provided in: https://www.rabbitmq.com/install-windows.html.

3. Enable RabbitMQ management by running the following command from the location: `<Rabbitmq installed dir>/sbin`

   ```
   rabbitmq-plugins.bat enable rabbitmq_management
   ```

4. Restart the RabbitMQ service.

5. If using localhost to access RabbitMQ, verify access by using the URL: `localhost:15672`

   The default `username:password` for the RabbitMQ sign in is `guest:guest`. This is a user account with Admin privileges, which is limited to localhost.

   Notes

   - If you intend to use localhost to access RabbitMQ, ignore the next step #6 and set the RMQ_HOST field value as `localhost` and use the default `username:password` in the properties file during the Intelligent Viewing installation.

- If you intend to use “IP address or FQDN” to access RabbitMQ instead of localhost, follow the next optional step #6.

6. Optional To access RabbitMQ using the “IP address or FQDN”, you must create a user account for remote host connections. To create the account, run the following commands from the location: `<Rabbitmq installed dir>/sbin`

```
rabbitmqctl.bat add_user <username> <password>
rabbitmqctl.bat set_user_tags <user_name> administrator
rabbitmqctl.bat set_permissions -p / <user_name> ".*" ".*" ".*"
```

For details, see https://www.rabbitmq.com/rabbitmqctl.8.html#User_Management.

a. After creating the account, verify that RabbitMQ is accessible using the URL: `<ipaddress/FQDN>:15672` and that you can sign in using the new account.

b. Set the RMQ_HOST field value as `<ipaddress/FQDN>` and use the new `username:password` in the properties file during the Intelligent Viewing installation.

Notes

- If RabbitMQ remains inaccessible, clear the browser cache or retry with the browser’s Incognito mode enabled.
- You must provide the RabbitMQ username and password when installing Intelligent Viewing.
- For additional troubleshooting tips, see “Installer error messages” on page 111.

3.3.2 Install database server

Install your preferred database server.

Supported databases and required versions are:

- **PostgreSQL** version 14 through 17
- **Oracle** version 19c (with the latest patch) through 21c
- **Microsoft SQL Server** 2019 and 2022

3.3.2.1 PostgreSQL

If PostgreSQL is your target database, then install a supported version: 14 through 17.

To download PostgreSQL, refer to https://www.postgresql.org/download/.

To install PostgreSQL:

1. Using **pgadmin** or the tool of your choice, create a blank database for the app, then add the `pgcrypto` extension by executing the following against the new database:

   ```
   create extension pgcrypto;
   ```

2. If PostgreSQL will be accessed from a separate host, you must allow the IP addresses in `pg_hba.conf`. See https://www.postgresql.org/docs/current/auth-pg-hba-conf.html.

   The following entry will allow all, but in a production environment, this should be limited only to the IP addresses that need access.

   ```
   "host all all <IP_Address>/32 md5"
   ```

> **!** **Privileges**
>
> **Creating the database**
> Before running the Intelligent Viewing installation, the PostgreSQL database must be created. The user creating the database must either be a super user or have the CREATEDB privilege. For more information, see https://www.postgresql.org/docs/current/sql-createdatabase.html.
>
> **Creating database objects**
> As part of the Intelligent Viewing installation and startup, the transformation services create the schema, tables, and indexes. To allow the creation of these required database objects, the database user that is set in the configuration file must have the CREATE privilege. For more information, see https://www.postgresql.org/docs/current/sql-createschema.html.

3.3.2.2 Microsoft SQL Server

If Microsoft SQL Server is your target database, then do the following.

To install SQL Server:

1. To download Microsoft SQL Server, refer to https://www.microsoft.com/en-in/sql-server/sql-server-downloads.

2. Connect to Microsoft SQL Server using SQL Server/Windows authentication and execute the following SQL commands.

3. To create a database for Intelligent Viewing:

```
IF NOT EXISTS(SELECT * FROM sys.databases WHERE name = 'iv')
CREATE DATABASE [iv]
USE [master]
```

4. To create a login for Intelligent Viewing at the server level to be used for logging into the iv database:

   ```
   IF NOT EXISTS(select loginname from master.dbo.syslogins where name = 'IVLOGIN' and
   dbname = 'iv')
   CREATE LOGIN [IVLOGIN] WITH PASSWORD=N'Password1!', DEFAULT_DATABASE=[iv],
   CHECK_EXPIRATION=OFF, CHECK_POLICY=ON;
   ```

   Note: If you want to use a single Microsoft SQL server instance, ignore this note. Otherwise, refer to these additional setup steps for the cluster environment.

   In a cluster environment, the LOGIN should be created in all nodes with one security identification number (SID). After adding the iv database to the cluster or high availability setup environment, only the database level objects (such as users and tables) are synchronized between the failover nodes and the primary node, but not the server level logins. You must create the server logins in all of the failover nodes as instructed below:

   - In the primary node, get the SID of the server login:

     ```
     select sid from sys.server_principals where name=N'IVLOGIN';
     ```

   - In all of the failover nodes, create the login with the above SID:

     ```
     USE [master]
     CREATE LOGIN [IVLOGIN] WITH PASSWORD=N'Password1!',
     SID=<sid_of_the_login_in_primary_replica>,
     DEFAULT_DATABASE=[iv], CHECK_EXPIRATION=OFF, CHECK_POLICY=ON;
     ```

5. To create a user at the database level that is mapped to the login:

   ```
   USE [iv]
   IF NOT EXISTS(SELECT * FROM sys.database_principals WHERE name = 'IVUSR')
   CREATE USER [IVUSR] FOR LOGIN [IVLOGIN];
   ```

6. The list of permissions that are required for users to perform actions in Intelligent Viewing are provided below:

   ```
   GRANT ALTER TO [IVUSR]
   GRANT CREATE FUNCTION TO [IVUSR]
   GRANT CREATE PROCEDURE TO [IVUSR]
   GRANT CREATE SCHEMA TO [IVUSR]
   GRANT CREATE TABLE TO [IVUSR]
   GRANT CREATE VIEW TO [IVUSR]
   GRANT DELETE TO [IVUSR]
   GRANT EXECUTE TO [IVUSR]
   GRANT INSERT TO [IVUSR]
   GRANT REFERENCES TO [IVUSR]
   GRANT SELECT TO [IVUSR]
   GRANT UPDATE TO [IVUSR]
   ```

   These commands create the iv database, the IVLOGIN login, and the IVUSR user.

   Users can provide the DB name, Login name, and password at their convenience, and the same DB name, Login name, and password must be provided while installing Intelligent Viewing.

Note: Intelligent Viewing uses Flyway migrator to import configuration details and to create the database objects (such as schema and table) for Intelligent Viewing users. Because Intelligent Viewing is used by Flyway, it must have permissions to perform this activity. The above commands grant the necessary permissions.

7. Run the following command to enable READ COMMITTED SNAPSHOT ISOLATION (RSCI). RSCI controls row versioning for the Read Committed isolation level and, when enabled, prevents read queries from being blocked by concurrent writes.

   ```
   ALTER DATABASE iv SET READ_COMMITTED_SNAPSHOT ON;
   ```

3.3.2.3 Oracle

If Oracle is your target database, then install a supported version: 19c (with the latest patch) through 21c.

To download Oracle, refer to https://www.oracle.com/in/database/technologies/oracle-database-software-downloads.html.

Upgrade note:

If you are migrating an Intelligent Viewing installation at version 24.2 or earlier to a more recent version, the Oracle database schema has changed starting in 24.3 and you must run the Oracle SQL script to migrate the data from the previous schema to the new schema.

This schema update script should be run independent of any other scripts that are executed during database maintenance. Because the script provided contains a COMMIT statement within it, if you are running any other scripts in succession, those scripts will also get committed into the database.

For more information, see this OpenText support article (https://support.opentext.com/csm?id=kb_article_view&sysparm_article=KB0811280).

Follow the steps below to create a Pluggable Database (PDB) in the Container Database (CDB):

1. Open **sqlplus** from a command line:

   ```
   sqlplus '/ as sysdba'
   ```

2. Verify that you are currently in the CDB root:

   ```
   SHOW CON_NAME;
   CON_NAME
   ------------------------------
   CDB$ROOT
   ```

3. Create a pluggable database:

   ```
   CREATE PLUGGABLE DATABASE <pdb_name> ADMIN USER <pdb_admin_username>
   IDENTIFIED BY <pd_password>
   FILE_NAME_CONVERT = ('<path to pbdseed directory>\pdbseed\', '/<pdb_name>/');
   ```

4. Connect to the pluggable database:

```
CONNECT <pdb_name> as sysdba;
password: <type_the_pdb_password>
ALTER SESSION SET CONTAINER = <pdb_name>;
```

5. Verify that the current connection is the newly created PDB:

```
SHOW CON_NAME;
```

6. Startup the PDB if it is not already open/started:

```
STARTUP;
```

7. Create a tablespace against the newly created PDB:

```
CREATE TABLESPACE iv_ts_space DATAFILE 'iv_ts_space_01.dbf' SIZE 10M
AUTOEXTEND ON MAXSIZE UNLIMITED;
```

8. Create a user for the PDB and grant privileges:

```
CREATE USER <username> IDENTIFIED BY <user_password> DEFAULT TABLESPACE iv_ts_space
TEMPORARY TABLESPACE TEMP;
ALTER USER <username> QUOTA UNLIMITED ON iv_ts_space;
GRANT CREATE TRIGGER,CREATE TABLE,CREATE PROCEDURE,CREATE SESSION TO <username>;
```

   In the above code, the values set for <username> and <user_password> should be used as the database user and password while installing Intelligent Viewing.

9. If you have already created the PDB as a test and want to restart from scratch, use the below commands to drop the pluggable database:

```
sqlplus '/ as sysdba'
ALTER PLUGGABLE DATABASE <pdb_name> CLOSE INSTANCES=ALL;
DROP PLUGGABLE DATABASE <pdb_name> INCLUDING DATAFILES;
```

Oracle Data Guard notes:

If you are using Oracle with Data Guard as your database connection, refer to the following support article for additional configuration details: https://support.opentext.com/csm?id=kb_article_view&sysparm_article=KB0784607.

For versions 24.3 and later, the following notes are appended:

- To connect to the dataguard, you must populate the PITHOS_HOST env variable in all of the transformation services (ConfigService, PublicationService, PublisherService) with comma separated hostnames of all the nodes in the dataguard.
- For the MarkupService to connect, locate the .env file at `service_name\.env` and populate the DB_HOST env variable with comma separated hostnames of all the nodes in the dataguard. Post provided values to the DB_HOST property in the .env file of the Markup Service and restart the service.

  There are no PITHOS_ORACLE_* env variables in the Markup Service as they are only used by the transformation services.
- Oracle Instant Client package setup is required only for Intelligent Viewing Versions prior to version 24.3.

3.3.3 Install OpenText Directory Services (OTDS)

OpenText™ Directory Service (OTDS) is required. For information on installing and configuring OTDS, see *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.

! **Important**

- Use the latest version of OTDS, which can be downloaded from OpenText My Support (https://knowledge.opentext.com/knowledge/llisapi.dll/Open/OTDS). Refer to the Intelligent Viewing release notes for the minimum version supported.
- The OTDS installer stops the Tomcat Service but does not restart it. Make sure that OTDS is running before installing Intelligent Viewing.
- If you are upgrading your Intelligent Viewing installation from a previous version, refer to the upgrade note regarding Oauth client and secret retention in "Run the Intelligent Viewing installer" on page 38.

There are multiple ways that you can configure the Intelligent Viewing services with OTDS:

1. Installing the services to separate environments (**Recommended**).

   If you plan to install the Intelligent Viewing services to separate environments (multiple installer runs), you should first run `OTDSConfig.exe` (which should only be run one time for the entire deployment). This will output the client IDs and secrets to a `.properties` file. See the next section "OTDS setup" on page 27 for usage details.

   The resulting `.properties` file should then be provided to `OTDS_AUTH_FILE_PATH` in "Installer properties" on page 40 each time the installer is run in each environment.

2. Installing all of the services on the same machine (**For demonstration**).

   If you provide the OTDS admin user and password, the installer will upload the Intelligent Viewing license, create all of the OAuth clients, and configure each of the Intelligent Viewing services to use the resulting clients and secrets. You do not need to populate any of the related AUTH client or secret properties if you choose this method. This option should only be chosen if you are installing all of the Intelligent Viewing services together because the license and all of the OAuth clients are created new whenever this setup runs. Running setup for one service installation, and then running it again for another service installation will result in the secrets used by the first service no longer being valid.

3. Manually specifying clientIDs and secrets.

   If your clientIDs and secrets are known, and you prefer to pass them at the command prompt or set them in the main `.properties` file that gets passed to `PROPERTIES_FILE_PATH`, you can set them individually by providing them to each services' related `<service>_AUTH_CLIENT_ID` and `<service>_AUTH_CLIENT_SECRET` variables.

3.3.3.1 OTDS setup

OTDSConfig.exe notes:

- You do not need to run the `OTDSConfig.exe` tool if you provide OTDS admin credentials to the installer. The installer will run it in that case.
- The file `INTELLIGENT_VIEWING.lic` must be in your working directory for this tool to succeed. If the tool was run without the license present in the same directory as `OTDSConfig.exe`, you must remove the users and resource before trying it again.
- Attempting to change the client prefix when the demo license is still in place will result in failure because the two seats that the license includes are already consumed by the users from the first run.

What this command prompt tool does:

- Uploads the Intelligent Viewing license to OTDS.
- Creates the resource.
- Creates the OAuth clients.
- Outputs a `.properties` file containing the LICENSE_RESOURCE secret as well as the OAuth clients and each of their secrets.

If this tool is run manually, either:

- The output file's path can be provided to the installer property `OTDS_AUTH_FILE_PATH`.
- The output file's contents can be merged into the main properties file that gets passed to `PROPERTIES_FILE_PATH`.
- The individual properties can be set at the installer command prompt.

Tool parameters

Run the `OTDSConfig.exe` tool with the following parameters:

```
OTDSConfig.exe -u OTDS_ORIGIN -a OTDS_ADMIN_USER -p OTDS_ADMIN_PASSWORD
```

Argument	Description
`?`	Outputs command-line options
`-u`	OTDS URL
`-a`	OTDS Admin User
`-p`	OTDS Admin Password
`-n`	(Optional) Prefix for output file names. Default is "_default_"
`-r`	(Optional) Prefix for resource name. Default is "iv"
`-f`	(Optional) Prefix for client names. Default is "iv"

Note: Running the tool with no arguments will display usage information.

Content Server parameters

The parameters below are required ONLY if installing for Content Server.

Parameter	Description
`--cs-url`	Content Server URL. For example, `http://localhost:8080/OTCS/cs`
`--cs-admin`	Content Server admin user name.
`--cs-password`	Content Server admin password.
`--cs-resource-name`	OTDS resource name for Content Server.
`--publication-url`	IV Publication Service URL.
`--search-url`	IV Search Service URL.
`--viewer-url`	IV Viewer Service URL.
`--markup-url`	IV Markup Service URL.
`--asset-url`	IV Asset Service URL.

If problems are encountered when running the OTDSConfig tool, specific exit codes are returned to indicate the problem.

3.4 Security settings

The Intelligent Viewing services communicate with each other through HTTP calls to REST APIs. In a production environment, these calls should be configured to use HTTPS, a more secure version of HTTP that verifies the identities of the server and the client and encrypts the communication between them to prevent outside interference. Intelligent Viewing supports HTTPS, but requires extra configuration to successfully setup the environment with this support. This extra configuration can be done at install time by providing appropriate values to the installation properties.

Intelligent Viewing uses TLS (which is sometimes generically referred to as SSL for historical reasons) to secure connections. The *server* and the *client* in a TLS connection is determined by the direction of the connection, not the functional software role. The software that is initiating the connection is acting as the client and the software that is accepting the connection is acting as the server. In Intelligent Viewing, the services can be a client or a server at different times when communicating internally with each other. Securing the connections between the services prevents outside interference by a network node that the communication is passing through. When external software is connecting to Intelligent Viewing services to make API requests, the services are acting as TLS servers. Securing the connection between external software and the Intelligent Viewing endpoints helps ensure that the API calls are being sent to the expected server and only that server. When the viewer is running in a web browser, it is functioning as external software in a client role and is using the web browser's trust settings to determine if it should connect to the Intelligent Viewing endpoints.

Software that is acting as a TLS server uses a server *certificate* to validate that the server is the intended server and to set up the end-to-end encryption of the connection. The certificate contains a *public key* that can decrypt data encrypted with a corresponding *private key*, which is separate from the certificate. When an incoming TLS connection is made over HTTPS, the certificate is provided to the connecting client.

Software that is acting as a TLS client must decide whether it should *trust* the certificate that the TLS server provides. Certificates are usually issued by a *Certificate Authority* (CA) and can be traced back to the issuing CA through a *certificate chain*, validating that each certificate in the chain was created with verifiable credentials. Software that supports TLS usually comes with a built-in set of certificates for the largest CAs. When the TLS server provides a certificate chain, the client will validate each certificate up the chain until it reaches one of the built-in certificates, at which point it will trust the incoming certificate.

IV services and TLS

The Intelligent Viewing services must be configured with both a server certificate (for when they are acting as a TLS server) and with appropriate trust settings (for when they are acting as a TLS client).

Intelligent Viewing services read both the server certificate and the corresponding private key from a *keystore* file when they start up. The keystore file can be read from either PKCS12 (often with a .pfx extension) or JKS (often with a .jks extension) format. Keystore files are protected by a *keystore password* and, if there is more than one certificate/private key pair in the file, they are each given a different *keystore alias*. In JKS keystores, each keystore alias can have its own *keystore alias password*. If there is only a single entry in the file, then the alias and alias password do not need to be provided.

Intelligent Viewing services use trust settings supplied by the environment that each service is running within. The Asset, Configuration, Publication, and Publisher Services run in Java and use the Java run-time's trust settings. The Viewer, Highlight, and Markup Services run in Node.js and use the Node.js run-time's trust settings. Each of these run-time environments provides reasonable Certificate Authority certificates that should allow most purchased TLS certificates to be validated and trusted. Each environment has its own method of adding additional trusted certificates if required.

TLS has had several versions of the protocol defined over its lifespan. Intelligent Viewing only supports version 1.2 (released in 2008) and version 1.3 (released in 2018). Earlier versions are deprecated as of 2021 and are no longer generally supported. Servers and/or clients that only support earlier versions of TLS will not work with the Intelligent Viewing TLS connections.

RabbitMQ and TLS

RabbitMQ can also be configured with TLS. An overview of TLS in RabbitMQ can be found here: https://www.rabbitmq.com/ssl.html. The Publication and Publisher Services act as TLS clients to the RabbitMQ server when TLS is enabled in RabbitMQ.

Database and TLS

Database access is also usually configured with TLS. The specifics of setting up the database for TLS access will depend on the database in use. Refer to the documentation provided with the database for the best security practices. The Config, Markup, Publication, and Publisher Services all act as TLS clients to the database server when TLS is enabled on the database.

3.4.1 Installer TLS settings

IV services

To enable TLS in the Intelligent Viewing services, the following properties should be set with appropriate values before running the installer:

- `HTTP_PROTOCOL` must be changed from `http` to `https`.
- `KEYSTORE_PATH` must be set to a valid filepath to a PKCS12 or JKS keystore file containing the certificate and private key that the IV services on that machine will use.
- `KEYSTORE_PASSWORD` must be set to the password for the file given in `KEYSTORE_PATH`.
- `KEYSTORE_ALIAS` can optionally be set if the configured keystore file has multiple certificates in it.

RabbitMQ

If the configured RabbitMQ instance is set up to require TLS, the following properties should also be set:

- `RMQ_USE_SSL` must be changed from `FALSE` to `TRUE`.
- `RMQ_KEYSTORE_PATH` must be set to a valid filepath to a PKCS12 or JKS keystore file containing the certificate and private key that the IV services on that machine will use when clients connect to it.
- `RMQ_KEYSTORE_PASSWORD` must be set to the password for the file given in `RMQ_KEYSTORE_PATH`.
- `RMQ_KEYSTORE_TYPE` should be set to either `PKCS12` or `JKS`, depending on the type of file given in `RMQ_KEYSTORE_PATH` (and optionally `RMQ_TRUSTSTORE_PATH`).
- `RMQ_TRUSTSTORE_PATH` can optionally be set to a valid filepath to a `PKCS12` or `JKS` keystore file containing the certificate of the RabbitMQ server. If the RabbitMQ server is signed by a CA that is trusted by the default truststore

for Java, this option is unnecessary. If the RabbitMQ server is using a self-signed certificate or an alternate CA, this parameter allows providing a separate truststore, rather than adding the RabbitMQ certificate to the default truststore.

Note: The provided file should also be of the type provided in `RMQ_KEYSTORE_TYPE`.

- `RMQ_TRUSTSTORE_PASSWORD` can optionally be set to the password for the file given in `RMQ_TRUSTSTORE_PATH`.

Database

If the configured database is set up to require TLS, the following properties should also be set:

SSL support note

SSL support is only available for the PostgreSQL database.

- `DEFAULT_DB_USE_SSL` must be changed from `FALSE` to `TRUE`, or use an equivalent `<other>_DB_USE_SSL` option. See the "Installer properties" on page 40 table and "Database settings" on page 87 for details and options.
- `CONFIG_DB_SSL_MODE` must be changed from `disable` to `prefer`, `require`, `verify-ca`, or `verify-full`.
- `PUBLISHER_DB_SSL_MODE` must be changed from `disable` to `prefer`, `require`, `verify-ca`, or `verify-full`.
- `PUBLICATION_DB_SSL_MODE` must be changed from `disable` to `prefer`, `require`, `verify-ca`, or `verify-full`.

When setting the database SSL mode values, the following options are permitted:

SSL mode	Description
disable	Disables TLS communication. Does not provide any security.
allow	Enables TLS communication if the database requires it but will still connect if the database doesn't support it.
prefer	Enables TLS communication if the database supports it but will still connect if the database doesn't support it.
require	Will not connect to the database unless the database supports TLS and uses it.
verify-ca	Will not connect to the database unless the database supports TLS and uses it, and the certificate used is issued by a trusted certificate authority (CA).

SSL mode	Description
verify-full	Will not connect to the database unless the database supports TLS and uses it, and the certificate used is issued by a trusted (CA), and the server host name matches the name in the certificate.

In most cases, the most secure option, `verify-full`, is the correct option to use when enabling TLS. In certain cases, it might be permissible to use a less secure option if the environment is otherwise secured.

3.4.2 Post-installation TLS settings

JKS-format keystore files support a separate password for each different aliased key in the keystore, unlike PKCS-format keystore files that use the same keystore password for all keys in the file. JKS-format keystore files are not recommended. However, if the system must be configured with a pre-existing JKS file with password-protected aliased keys, it might require setting the special alias password. The environment variable KEYSTORE_ALIAS_PASSWORD can be used to set this password. This variable is found in the .env/env.conf configuration files in each service installation folder that sets the environment variables for the service. The default setting is an empty entry. Changes to these files require the service to be restarted to pick up the change.

Intelligent Viewing supports versions 1.2 and 1.3 of TLS. In some situations, it might be desirable to force Intelligent Viewing to only make specifically 1.2 or 1.3 connections. The IV services support setting the minimum and maximum version to allow. The environment variables TLS_MIN_VERSION and TLS_MAX_VERSION can be set to `TLSv1.2` or `TLSv1.3`. The default is a minimum version of 1.2 and a maximum version of 1.3. Setting both variables to `TLSv1.3` will force the service to only make 1.3 connections. These variables are found in the .env/env.conf configuration files in each service installation folder that set the environment variables for the service. After changes to these files, the service must be restarted to pick up the change.

3.4.3 Troubleshooting TLS

Misconfigured TLS can manifest in several different ways. Services may fail to start entirely if they cannot make a connection to a necessary resource, or services might start successfully but be unable to complete REST API calls. In viewing situations, the viewer might load, but be unable to display specific types of content.

- If services are failing to start, the service logs should be reviewed for “Network Error” errors. These can occur if the service is configured with an http URI for another service, but that other service is running on an https service, or the reverse. They can also occur if a service is running as https, but the connecting service does not trust the certificate in use.
- If the Viewer Service is configured with TLS and the certificate it presents is not trusted by the browser, the viewer will not load from the Viewer Service. An integrating web page that is attempting to reach the Viewer Service through the

https URI can fail quietly with a Cross-Origin Resource Sharing (CORS) error that is not reported directly to the user. The viewer will fail to appear, and the CORS error will be reported in the browser's console.

- If the viewer is successfully loaded, but the Highlight Service or Markup Service is configured with TLS and the certificates that they present are not trusted by the browser, the viewer will be unable to load bookmarks, search text, or annotations. The REST API calls in the viewer will fail with a "Network Error" message, indicating that they cannot read from the other service.
- The viewing services use Node.js functions to validate the certificate. If the services are set up with IP address URLs (for example, `https://xxx.xxx.xxx.xxx:<port>`) instead of FQDN URLS (`https://computer.domain.com:<port>`) the server certificate must include the IP address in the `Subject Alternate Names` field. If the server certificate only includes the IP address in the `Common Name field`, Node.js will reject the connection with a "ERR_TLS_CERT_ALTNAME_INVALID" message.

3.4.4 Using a self-signed certificate

A production system using TLS should always use a properly signed and validated PKI certificate that is trusted by a Certificate Authority. For testing or demonstration purposes it can be expensive or time-consuming to acquire a fully trusted certificate. A self-signed certificate is one that only claims itself as the signing authority, rather than having a certificate chain back to a trusted CA. See the following section "Creating and installing a self-signed certificate for testing and demonstration" on page 33 for instructions on how to generate a self-signed certificate and install it as trusted on a specific machine. Note that other machines connecting to that machine will also need to be set up to trust the certificate to connect to the test/demo machine.

3.4.5 Creating and installing a self-signed certificate for testing and demonstration

This section provides instructions for setting up a machine or machines with a self-signed certificate for the purposes of testing Intelligent Viewing with a TLS setup. See "Security settings" on page 28 for background information on using TLS with Intelligent Viewing and terminology.

! **Important**

Self-signed certificates should not be used in a production environment unless a security expert is managing their use.

3.4.5.1 Overview

The process of utilizing a self-signed certificate for Intelligent Viewing consists of three basic operations before the Intelligent Viewing installer is run:

1. A suitable certificate must be created.
2. A keystore containing that certificate must be created.
3. Each component that will be asked to trust that new certificate must be made aware of the existence of the certificate.

These operations can be accomplished by different tools and exact commands can vary between toolsets, format choices, and platforms. The examples below show commands using the `keytool` command that is part of a Java installation. Another valid toolset choice would be the OpenSSL toolkit (https://www.openssl.org/).

3.4.5.2 Certificate creation

The Java keytool provides a command to generate a public/private keypair and certificate and directly store it into a keystore. A minimal command to generate a suitable keypair into a file is:

```
keytool -genkeypair -keyalg RSA -keystore <file> -ext "SAN=dns:<host>,ip:<host ip
address>"
```

The *`keyalg`* argument selecting the RSA key algorithm overrides the default DSA algorithm, which is no longer recommended. The *`keystore`* argument selects the output file to be created. The *`ext`* argument supplies X.509 certification extension parameters. The specific parameters given here will set the Subject Alternate Name (SAN) values for *`DNSName`* (dns:<host>) and *`IPAddress`* (ip:<host ip address>) for the server into the certificate. These values help the TLS connection verify that the certificate belongs to the server using it.

Running the command will prompt for a password for the keystore file and a name, organizational unit, organization, locality, state, and country code. The server name should be provided at the prompt for first and last name. This field is the `Common Name` of the certificate. The common name should be a fully qualified domain name (FQDN) and should match the <host> given to the *`ext`* parameter. Node.js will not recognize connections made to an IP if the IP is provided only in the common name. The IP must be provided in the SAN value for *`IPAddress`* to be recognized.

The certificate created with this minimal command will be valid for 90 days. A *`validity`* argument can be provided to explicitly set the number of days the certificate should be valid.

The resulting keystore will be a PKCS12 format keystore, the default type in Java as of Java 9 and an industry-standard format. It can be examined with the command:

```
keytool -list -v -keystore <filename>
```

This command will list (*`-list`*) the contents of the keystore (*`-keystore`*) with a verbose output (*`-v`*). The verbose output provides more information in the listing and is needed to confirm the `Subject Alternate Name` extension fields.

3.4.5.3 Keystore creation

When using the Java keytool program to generate a certificate, the keystore creation happens in the same step. If another tool is being used to generate the certificate or multiple certificates need to be stored into the same keystore, the Java keytool program can import certificates or entire keystores. See the keytool documentation for details.

If using another tool, it is recommended to create a PKCS12 format keystore if possible. JKS format keystores are supported but not recommended.

3.4.5.4 Truststore installation

Servers using self-signed certificates will not be immediately trusted by clients connecting to them through TLS because the self-signed certificates do not have a certificate chain going back to a trusted Certificate Authority. To allow clients to connect, the client machines must be explicitly given the public certificate from the self-signed certificate. The public certificate needs to be separated from the keystore that contains the private key. Putting the entire keystore including the private key on the client machine would negate the security of having a secret private key. To export just the public certificate from the keystore, use the following command:

```
keytool -exportcert -keystore <file> -rfc -file <certificate file>
```

This command will export the public certificate from a keystore file (*`file`*) into an RFC-style (*`-rfc`*), text-based encoded certificate file (*`certificate file`*). The format of the data in the file will be PEM but naming the file with a CRT extension (for example, `iv-server.crt`) is recommended so that OS tools will recognize it as a certificate.

Java

On each Intelligent Viewing machine that is running a transformation service, the certificate must be added to the certificates that Java trusts. This is a special keystore (truststore) installed with Java. The default password for this keystore, on a clean installation of the Java Runtime Environment, is `changeit`. This password can be used to install the self-signed certificate. The command to install into the Java truststore is:

```
keytool -import -cacerts -file <certificate file>
```

This command will prompt for the `changeit` password, display the self-signed certificate, and prompt whether the certificate should be trusted.

Node.js

On each Intelligent Viewing machine that is running a viewing service, the certificate needs to be added to the certificates that Node.js trusts. Node.js will read a filepath from the environment variable NODE_EXTRA_CA_CERTS and read a PEM-formatted file of certificates to trust from that location. The certificate file (or a copy of the certificate file) should be accessible to each service.

The NODE_EXTRA_CA_CERTS environment variable can be a System environment variable or a user environment variable for the account that the service is running under. Each viewing (Node.js) service must be restarted after creating or changing the NODE_EXTRA_CA_CERTS environment variable.

Web browser

Client machines that will be connecting to Intelligent Viewing servers that are using a self-signed certificate must be configured to trust that certificate. This process varies from browser to browser and operating system to operating system, and sometimes security policies prevent the installation of extra certificates.

Windows Client Machines

The PEM-format certificate key can be directly installed into Window's Trusted Root Certification Authorities truststore. To do this, right-click the file and click **Install Certificate** (ensure the file is named with a .crt or .cer extension). Select **Current User**, choosing to select the store, then use the **Browse** button to select the **Trusted Root Certification Authorities** certificate store. This will automatically be used by most Windows web browsers.

Linux Client Machines

Linux applications that use TLS typically look in specific files used by the OpenSSL environment. The PEM-format certificate file with a .CRT extension can be installed into the OpenSSL environment by using a system-provided Linux utility. In some distributions, this will be the *update-ca-certificates* tool that is part of the *ca-certificate* package. In other distributions, the *update-ca-trust* tool that is part of the *ca-certificates-utils* package does the same task. Consult the documentation for the appropriate tool for the distribution in use for installation instructions. The OpenSSL environment will automatically be used by most Linux web browsers.

3.4.5.5 Troubleshooting self-signed certificates

See "Troubleshooting TLS" on page 32 for generic TLS troubleshooting. Self-signed certificates can additionally show errors about "DEPTH_ZERO_SELF_SIGNED_CERT" and "Invalid request: self signed certificate", as well as the more generic "Network Error". These errors indicate that the server is correctly using TLS and supplying the self-signed certificate, but the connecting client is not configured to trust that certificate.

Chapter 4

Intelligent Viewing installation

This section provides information for installing OpenText Intelligent Viewing using a Windows installer.

Intelligent Viewing can be installed as a high availability solution. For information, see *OpenText Intelligent Viewing - High Availability Install Guide (CLIVSA-IHA)*.

4.1 Upgrade notes

- Customers upgrading to Intelligent Viewing CE 25.4 from version 21.3 or older must uninstall the existing version before running the new installer. Be sure to back up any desired files from the install directory, then uninstall from **Apps & features** before running the installer.
- If you are upgrading from version 21.4 or newer, there are no special parameters for running an upgrade. Simply run the new installer as if you are performing a new installation.
- Retaining Oauth client and secret details: If you have run OTDS config in a previous version and have generated an output file (`.properties`) containing a resource id, Oath clients, and secrets (which were added to the `installer.properties` file either by providing the path of the generated output file to the variable OTDS_AUTH_FILE_PATH or by providing the client and secret details individually for the required services), then those same details must be added into the `installer.properties` file of the new version that you are upgrading. Oauth clients that were created in the previous version will be used by the installer during the upgrade, so you should not run the OTDS config again until after the upgrade is completed.
- Installation of a new version will automatically archive old log files in the same location using the folder format [service name].archive-[yyyyMMddHHmmss]. This prevents potential permission issues caused by logs created by different user accounts in previous versions. Post-installation changes will not automatically adjust log files. It is therefore important to delete any log files using the same log location, as older logs may be owned by a different user and could prevent service startup or logging (See “Running Services under a custom account” on page 63).

4.2 Accounts used by Intelligent Viewing services

The Intelligent Viewing Services are installed under several built-in Windows accounts.

- **Asset Service** and **Publication Service** run as **Network Service**. These services are required for accessing the artifact cache if it is hosted on a different machine.
- **Publisher Service** and **Viewer Service** run as **Local System**. These services require the elevated privileges of the Local System account.
- **Configuration Service**, **Highlight Service**, and **Markup Service** run as **Local Service**. These services require minimal privileges.

Post installation, it is possible to run the services under another account (See "Running Services under a custom account" on page 63).

4.3 Run the Intelligent Viewing installer

Note: Any folder, whether created manually or by the installer, that the Intelligent Viewing services need to use must have the correct access permissions granted to the users accounts running those services.

Installation steps

1. Download the Intelligent Viewing CE 25.4 Windows installer from OpenText™ MySupport.

   https://support.opentext.com/ My Products:

   (Intelligent Viewing <*version*> > intelligent-viewing-packaging-windows-<*version*>.zip)

   The following files are included in the download package:

   - `IntelligentViewing_<buildnumber>.exe`: This is the installer package.
   - `IntelligentViewing_MSI.properties`: This file contains the MSI property values.
   - `OTDSConfig.exe`: See "OTDS setup" on page 27
   - `INTELLIGENT_VIEWING.lic`: This license file must be in your working directory if running `OTDSConfig.exe`.

   A `manifests` folder is also included in the installation package that contains information about the component versions.

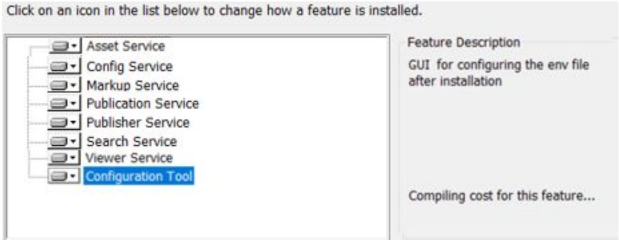
   An optional `ConfigurationTool` folder is included that provides an .exe file for configuring the environment files after installation.

2. Configure each of the values in the `IntelligentViewing_MSI.properties` file for your environment, and determine whether the installer will be performing setup of the OTCS/OTDS clients and the license resource. See the descriptions for OTDS_ADMIN_USER and OTCS_ADMIN_USER in the “Installer properties” on page 40 table for important notes. Important properties to verify include OTDS_ORIGIN, OTDS_ADMIN_PW, and DB_NAME; particularly if you are installing within a multi-tenant OTDS environment.

   Note: By default, Intelligent Viewing services do not send an authorization header in the http request when accessing source files at http URLs. If the source files being accessed by the services are at http (as opposed to https) URLs and an authorization header is required for access, the PUBLICATION_TRUSTED_SOURCE_ORIGINS and PUBLISHER_TRUSTED_SOURCE_ORIGINS properties must be updated with a comma delimited list of all origins where source files can be access from.

3. Run the Intelligent Viewing installer `IntelligentViewing_<buildnumber>.exe` and follow the prompts.

   - The feature installation page allows you to choose which services to install.



Click on an icon in the list below to change how a feature is installed.

- Asset Service
- Config Service
- Markup Service
- Publication Service
- Publisher Service
- Search Service
- Viewer Service
- Configuration Tool

Feature Description

GUI for configuring the env file after installation

Compiling cost for this feature...

   - During the installation you will need to specify a location of the MSI property values file `IntelligentViewing_MSI.properties`.

     See the description for PROPERTIES_FILE_PATH in “Installer properties” on page 40.

   - The **Configuration Tool** feature installs a `ConfigurationTool` folder and a desktop shortcut that provides a user interface for configuring Intelligent Viewing settings at any time after installation. See “Using the Configuration Tool“ on page 69.

     When the installation completes, you can select to **Launch the Windows Installer Configuration Tool** upon clicking **Finish**.

MSI installer logs

MSI installer logs are automatically created in `%temp%` for the current user. If the installer is running in UI mode, specific messages will display to indicate the problem encountered. Errors are logged to the MSI log regardless. When the installation completes, you can select to **Show the Windows installer log** upon clicking **Finish**.

4.4 Installer properties

Installation properties can be set at the command prompt individually as standard MSI properties, but the majority are expected to be provided to the installer in a plain text file. The text file contains `PROPERTY=value` pairs, one property per line. A basic file, `IntelligentViewing_MSI.properties`, is provided with defaults. Values provided in this file take precedence over any values provided at the command prompt. This file supports comments, where the line begins with the “#” character. Most of the properties provided are passed through to the correct environment files, with the correct names, for the appropriate services. These files can be found in `<service>/config/env.conf` for the Java-based services and `<service>/.env` for the JavaScript-based services.

Corrections or adjustments can be made to the environment files at any time after installation using the Configuration Tool. See “Using the Configuration Tool“ on page 69. If you will be using the tool, set the MSI property ALLOW_DEFERRED_CONFIGURATION to `true`, which will allow you to skip setting the optional values in the MSI file.

The default properties file loaded will be from `[SETUPEXEDIR]\IntelligentViewing_MSI.properties`. You can specify a different location by passing it in the MSI property PROPERTIES_FILE_PATH at the command prompt. If running in UI mode, a dialog will prompt you for this file.

INSTALLDIR

Description: Base directory to install the application into. Services will be installed in subdirectories beneath this directory.

Default: Default value is not provided in the file, but defaults to `C:\Program Files\OpenText\Intelligent Viewing`

Used by feature(s): All

ADDLOCAL

Description: Specifies features to be installed. Example:
`ADDLOCAL="AssetService,ConfigService,MarkupService,PublicationService,PublisherService,SearchService,ViewerService"`

Default: Blank

Used by feature(s): All

ALLOW_DEFERRED_CONFIGURATION

Description: Determines whether the installer will skip configuration and validation of optional settings, and service startup. Set this property to `true` if you will be installing and using the Configuration Tool to configure the environment files after installation. Settings that are not mandatory will be ignored during installation and are expected to be configured afterwards through the tool’s user interface. The installer will skip validation and will not start the services.

Settings that are updated post-installation with the Configuration Tool are automatically validated. You can also control service startup and download log files within the application.

If you are not installing the Configuration Tool, using `false` will force all properties to be applied at the time of installation. If set to `false`, you must enter all of the required properties, for which the installer will validate the values and automatically start the services. User will still have the option to launch the configuration tool for any upgrade scenarios, log monitoring, and health checks. If this property is not included in the MSI file, `false` is used. If incorrect values are entered or mandatory properties are skipped, the installation will fail.

Default: `false`

Used by feature(s): All

HTTP_PROTOCOL

Description: Protocol to use for communication to the various hosts. Can be `http` or `https`.

This property is not included in the supplied `.properties` file, so it must be added there or, if perming a silent install, at the command prompt. If installing in GUI mode, the installer will prompt for this and the related username property values.

Default: `http`

Used by feature(s): All

ENABLE_CREDENTIAL_ENCRYPTION

Description: When `true`, values for sensitive properties are encrypted. Set to `false` to disable encryption. Missing or invalid values default to `true`.

Encrypted properties include:

```
KEYSTORE_PASSWORD
KEYSTORE_ALIAS
RMQ_USER
RMQ_PWD
RMQ_KEYSTORE_PASSWORD
RMQ_TRUSTSTORE_PASSWORD
CONFIG_DB_USER
CONFIG_DB_PWD
MARKUP_DB_USER
MARKUP_DB_PWD
PUBLISHER_AUTH_CLIENT_ID
PUBLISHER_AUTH_CLIENT_SECRET
PUBLISHER_DB_USER
PUBLISHER_DB_PWD
PUBLICATION_AUTH_CLIENT_ID
PUBLICATION_AUTH_CLIENT_SECRET
PUBLICATION_DB_USER
PUBLICATION_DB_PWD
SEARCH_AUTH_CLIENT_ID
SEARCH_AUTH_CLIENT_SECRET
MARKUP_JWT_SECRET
LICENSE_RESOURCE
```

Encrypted installer properties are added to the `.env/env.conf` files corresponding properties as `"encrypted:<encrypted value>"`.

Encrypted properties never have their values logged (even if the values are not encrypted).

Secret Properties values are not logged, and are not encrypted because they do not go into configuration files:

```
    OTDS_ADMIN_USER
    OTDS_ADMIN_PW
    DEFAULT_DB_USER
    DEFAULT_DB_PWD
```

Encryption is possible only through the installer. Manual adjustments of encrypted properties are not allowed after installation and a re-installation of IV is required for changes.

Default: `true`

Used by feature(s): All

KEYSTORE_PATH

Description: File path to a Java keystore used for enabling SSL. Only used if HTTP_PROTOCOL is set to `https`. For more information about SSL, see "Security settings" on page 28.

Default: Blank

Used by feature(s): AssetService, ConfigService, PublicationService, MarkupService, SearchService, ViewerService

KEYSTORE_PASSWORD

Description: Password for the Java keystore provided in KEYSTORE_PATH.

Default: Blank

Used by feature(s): AssetService, ConfigService, PublicationService, MarkupService, SearchService, ViewerService

KEYSTORE_ALIAS

Description: The alias name of the certificate to be used when multiple certificates exist in a keystore.

Default: Blank

Used by feature(s): AssetService, ConfigService, PublicationService, MarkupService, SearchService, ViewerService

ENFORCE_CORS_ORIGINS

Description: Value can be `true` or `false`. When `true`, the services return an `Access-Control-Allow-Origin` header and the installer adds the integration origin to the env variable, CORS_ORIGINS_LIST.

Default: `false`

Used by feature(s): AssetService, ConfigService, PublicationService, MarkupService, SearchService, ViewerService

CORS_ADDITIONAL_HEADERS_LIST

Description: A comma-separated list of header names to allow in addition to the default headers allowed in CORS requests.

Default: Blank

Used by feature(s): AssetService, ConfigService, PublicationService, MarkupService, SearchService, ViewerService

ENFORCE_FORWARDED_HOSTS

Description: Enable or disable forwarded host whitelist enforcement. If `true`, hostnames in HTTP `Forwarded` and `X-Forwarded-Host` request headers must appear in the list provided in FORWARDED_HOSTS_LIST.

Default: `false`

Used by feature(s): AssetService, PublicationService, MarkupService, SearchService, ViewerService

FORWARDED_HOSTS_LIST

Description: A comma-separated list of forwarded host strings.

Default: Blank

Used by feature(s): AssetService, PublicationService, MarkupService, SearchService, ViewerService

ENABLE_OAUTH

Description: Value can be `true` or `false`.

Default: `true`

Used by feature(s): AssetService, ConfigService, PublicationService, PublisherService

DEFAULT_HOST

Description: The DEFAULT_HOST value will be used as the value for each of the other *_HOST values, if those values are left blank. This allows installing all or most of the services on the same host by providing a single host in this value. Services that will be located on other hosts should override the DEFAULT_HOST with the appropriate x_HOST value. Value should be a full domain name, without any protocol or port (for example, `server.domain.com`). The hostname will be provided during viewing so that viewers can access resources, so the value should not be localhost, a non-routable IP, or an unqualified server name.

Default: Default initial value is `<fully-qualified-domain-name>`, which is invalid and must be customized.

Used by feature(s): AssetService, PublicationService, MarkupService, SearchService, ViewerService

DEFAULT_DB_HOST

Description: The DEFAULT_DB_HOST value will be used as the value for each of the other *_DB_HOST values, if those values are left blank. If individual services are using separate database parameters, those parameters can be overridden with the service-specific `x_DB_x` parameter.

Default: Default initial value is `<fully-qualified-domain-name>`, which is invalid and must be customized.

Used by feature(s): ConfigService, MarkupService, PublicationService, PublisherService

DEFAULT_DB_PORT

Description: The DEFAULT_DB_PORT value will be used as the value for each of the other *_DB_PORT values, if those values are left blank. If individual services are using separate database parameters, those parameters can be overridden with the service-specific `x_DB_x` parameter.

Default: Default initial value is `<databaseserver-port>`, which is invalid and must be customized.

Used by feature(s): ConfigService, MarkupService, PublicationService, PublisherService

DEFAULT_DB_NAME

Description: The DEFAULT_DB_NAME value will be used as the value for each of the other *_DB_NAME values, if those values are left blank. If individual services are using separate database parameters, those parameters can be overridden with the service-specific `x_DB_x` parameter.

Default: Default initial value is `<iv>`.

Used by feature(s): ConfigService, MarkupService, PublicationService, PublisherService

DEFAULT_DB_USER

Description: The DEFAULT_DB_USER value will be used as the value for each of the other *_DB_USER values, if those values are left blank. If individual services are using separate database parameters, those parameters can be overridden with the service-specific `x_DB_x` parameter.

Default: Default initial value is `<databaseserver-user>`, which is invalid. This value must be customized to the username you have set up to access your database.

Used by feature(s): ConfigService, MarkupService, PublicationService, PublisherService

DEFAULT_DB_PWD

Description: The DEFAULT_DB_PWD value will be used as the value for each of the other *_DB_PWD values, if those values are left blank. If individual services are using separate database parameters, those parameters can be overridden with the service-specific `x_DB_x` parameter.

Default: Default initial value is `<databaseserver-password>`, which is invalid. This value must be customized to the password you have chosen for your database.

Used by feature(s): ConfigService, MarkupService, PublicationService, PublisherService

DEFAULT_DB_USE_SSL

Description: The DEFAULT_DB_USE_SSL value will be used as the value for each of the other *_DB_USE_SSL values, if those values are left blank. If individual services are using separate database parameters, those parameters can be overridden with the service-specific `x_DB_x` parameter.

Default: `false`

Used by feature(s): ConfigService, MarkupService, PublicationService, PublisherService

SSL support note

SSL support is only available for the PostgreSQL database.

DEFAULT_DB_PROVIDER

Description: The DEFAULT_DB_PROVIDER will be used as the value for each of the other *_DB_PROVIDER values, if those values are left blank. If individual services are using separate database parameters, those parameters can be overridden with the service-specific x_DB_PROVIDER parameter. Valid values are `postgresql`, `mssql`, or `oracle`.

Default: `postgresql`

Used by feature(s): ConfigService, MarkupService, PublicationService, PublisherService

RMQ_HOST

Description: Hostname or IP of the RabbitMQ server.

Default: Default initial value is `<rabbitmqserver-host>`, which is invalid and must be customized.

Used by feature(s): PublicationService, PublisherService

RMQ_PORT

Description: AMQP port for RabbitMQ.

Default: `5672`

Used by feature(s): PublicationService, PublisherService

RMQ_USER

Description: RabbitMQ user.

Default: Default value is `<rabbitmqserver-user>`, which is invalid. This value must be customized.

Used by feature(s): PublicationService, PublisherService

RMQ_PWD

Description: RabbitMQ password.

Default: Default value is `<rabbitmqserver-password>`, which is invalid. This value must be customized.

Used by feature(s): PublicationService, PublisherService

RMQ_USE_SSL

Description: Should be set to `true` if the RabbitMSQ server is configured to use SSL (TLS), `false` otherwise.

Default: `false`

Used by feature(s): PublicationService, PublisherService

RMQ_KEYSTORE_TYPE

Description: Type of key used in the RabbitMQ keystore. Can be `JKS` or `PKCS12`.

Default: Blank

Used by feature(s): PublicationService, PublisherService

RMQ_KEYSTORE_PATH

Description: Path to keystore file to use with RabbitMQ. Only used if RMQ_USE_SSL is set to `true`. Note that the keystore file must be accessible to the user that the services will run under.

Default: Blank

Used by feature(s): PublicationService, PublisherService

RMQ_KEYSTORE_PASSWORD

Description: Password for the keystore file provided in RMQ_KEYSTORE_PATH.

Default: Blank

Used by feature(s): PublicationService, PublisherService

RMQ_TRUSTSTORE_PATH

Description: Path to truststore file to use with RabbitMQ. Only used if RMQ_USE_SSL is set to `true`. Note that the truststore file must be accessible to the user that the services will run under.

Default: Blank

Used by feature(s): PublicationService, PublisherService

RMQ_TRUSTSTORE_PASSWORD

Description: Password for the truststore file provided in RMQ_TRUSTSTORE_PATH.

Default: Blank

Used by feature(s): PublicationService, PublisherService

ARTIFACT_VOLUMES_ROOT

Description: Path to a local or shared folder where artifacts should be stored.

If a local path is given, then the installer creates this folder if it doesn't already exist. An uninstall will not remove this folder or any of the artifacts it contains.

If a shared path is given, then the folder must already exist. If the folder is hosted on a different server than the installed Publisher, Publication and/or Asset services, ensure that the IV server(s) and the ARTIFACT_VOLUMES_ROOT server are part of the same domain, or that they belong to domains with properly configured trust relationships.

Note: The shared folder must grant Read/Write/Execute (R/W/E) permissions to each IV server's domain computer account (DOMAIN \HOSTNAME$) through both sharing and NTFS permissions.

This configuration ensures that Intelligent Viewing services can reliably access and manage artifacts across machines without permissions issues.

Default: `C:\IVArtifacts`, which should be customized.

Used by feature(s): AssetService, PublicationService, PublisherService

OTDS_ORIGIN

Description: Full OTDS URL. For example, `http://otds.domain.com/otdsws`

When installing within a multi-tenant OTDS environment, you must set this property using the following syntax: `http(s)://<OTDSFQDN>:<PORT>/otdsws/otdstenant/<TENANT-ID>`

Default: Default initial value is `http://<host>:8080/otdsws`, which is invalid and must be customized.

Used by feature(s): All

OTDS_ADMIN_USER

Description: Admin user name for OTDS. If provided, the installer will upload the license to OTDS, create the OAuth clients, and configure the services to use the resulting clients and secrets.

Provide if installing all of the IV services together. Providing this will trigger the installer to run `OTDSConfig.exe` and handle OTDS client and license configuration for all of the services. If installing IV services separately, you should instead run `OTDSConfig.exe` manually once and then add the OAuth users, secrets, and resource value to each of your `.properties` files or pass the path to the output properties file using the OTDS_AUTH_FILE_PATH property.

When installing within a multi-tenant OTDS environment, you must set: `OTDS_ADMIN_USER=Tenant Admin User`

Default: Blank

Used by feature(s): All

OTDS_ADMIN_PW

Description: Admin password for OTDS. See notes above for OTDS_ADMIN_USER.

When installing within a multi-tenant OTDS environment, you must set: `OTDS_ADMIN_PW=Tenant Admin password`

Note: When running `OTDSConfig.exe` from the command prompt, if the password contains any special characters (such as $%^&), the password must be enclosed in double quotes.

Default: Blank

Used by feature(s): All

OTDS_RESOURCE_NAME

Description: Overrides the default resource name “iv” when the installer is running `OTDSConfig.exe`.

Default: Not set

Used by feature(s): All

OTDS_AUTH_CLIENT_PREFIX

Description: Overrides the default client name prefix “iv” when the installer is running `OTDSConfig.exe`.

Default: Not set

Used by feature(s): All

OTDS_AUTH_FILE_PATH

Description: Complete absolute filepath path to an OTDS configuration properties file created by `OTDSConfig.exe`. This cannot be just the filename or relative path; it must be a full absolute path.

If installing services on different machines, then `OTDSConfig.exe` should be run once manually and the properties file it creates can be provided to each installer so that the values are consistent between all services. Instead of manually merging the OTDS properties file with the installer properties file, you can provide the file directly with this path, and it will automatically add the properties from the OTDS file during the install.

If installing all services on the same machine, the installer can run `OTDSConfig.exe` directly and this property can be left unset. See OTDS_ADMIN_USER.

Default: Not set

Used by feature(s): PublicationService, PublisherService, SearchService

LICENSE_RESOURCE

Description: Base64 encoded license resource string from OTDS.

Default: Blank

Used by feature(s): All

ASSET_SERVICE_HOST

Description: Host name or IP for the Asset Service. See description of DEFAULT_HOST.

Default: Blank

Used by feature(s): AssetService, PublicationService, PublisherService

ASSET_SERVICE_PORT

Description: Port used for the Asset Service.

Default: `3350`

Used by feature(s): AssetService, PublicationService, PublisherService

CONFIG_SERVICE_HOST

Description: Host name or IP for the Config Service. See description of DEFAULT_HOST.

Default: Blank

Used by feature(s): ConfigService

CONFIG_SERVICE_PORT

Description: Port for the Config Service.

Default: `3351`

Used by feature(s): ConfigService

CONFIG_DB_HOST

Description: Host name or IP for the database server that the Config Service should use. See description of DEFAULT_DB_HOST.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_PORT

Description: Port for the database server that the Config Service should use. See description of DEFAULT_DB_PORT.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_NAME

Description: Name of the database that the Config Service should use. See description of DEFAULT_DB_NAME.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_USER

Description: Database user that the Config Service should use. See description of DEFAULT_DB_USER.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_PWD

Description: Database password that the Config Service should use. See description of DEFAULT_DB_PWD.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_USE_SSL

Description: Set to `true` if the database server is configured to use SSL, `false` otherwise. See description of DEFAULT_DB_USE_SSL.

Default: Blank

Used by feature(s): ConfigService

SSL support note

SSL support is only available for the PostgreSQL database.

CONFIG_DB_SSL_MODE

Description: If CONFIG_DB_USE_SSL is `false`, set to `disable`. If CONFIG_DB_USE_SSL is `true`, can be set to `prefer`, `require`, `verify-ca`, or `verify-full` in ascending order of security.

Default: `disable`

Used by feature(s): ConfigService

! **Important**
The following CONFIG_DB_<> settings are required when CONFIG_DB_SSL_MODE is set to either `verify-ca` or `verify-full` and DB_PROVIDER is set to PostgreSQL.

CONFIG_DB_PROVIDER

Description: Name of the database server that the Config Service should use. See description of DEFAULT_DB_PROVIDER.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_KEYSTORE_TYPE

Description: Type of key used in the database keystore. Can be `JKS` or `PKCS12`. Mandatory while using `verify-ca` or `verify-full` SSL mode.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_TRUSTSTORE_PATH

Description: Path to truststore file to use with the database. Mandatory while using `verify-ca` or `verify-full` SSL mode.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_TRUSTSTORE_PASSWORD

Description: Password for the truststore file provided in CONFIG_DB_TRUSTSTORE_PATH. Mandatory if the truststore is secured with any password.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_TRUSTSTORE_ALIAS

Description: Alias assigned to the certificate or key entry within your truststore. Add only if your trustrore has an alias.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_TRUSTSTORE_ALIAS_PASSWORD

Description: Password associated with the specified alias in your truststore Add only if your truststore has a password secured alias.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_KEYSTORE_PATH

Description: This optional setting is the path to the keystore file to use with the database.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_KEYSTORE_PASSWORD

Description: Password for the keystore file provided in CONFIG_DB_KEYSTORE_PATH.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_KEYSTORE_ALIAS

Description: Alias assigned to the certificate or key entry within your keystore. Add only if your keystore has an alias.

Default: Blank

Used by feature(s): ConfigService

CONFIG_DB_KEYSTORE_ALIAS_PASSWORD

Description: Password associated with the specified alias in your keystore Add only if your keystore has a password secured alias.

Default: Blank

Used by feature(s): ConfigService

MARKUP_SERVICE_HOST

Description: Hostname or IP for the database Markup Service to use. See description of DEFAULT_DB_HOST.

Default: Blank

Used by feature(s): MarkupService, PublisherService

MARKUP_SERVICE_PORT

Description: Port used for the Markup Service.

Default: `3352`

Used by feature(s): MarkupService, PublisherService

ENABLE_ROLE_BASED_ACCESS_CONTROL

Description: When `true`, the Markup Service requires an additional header X-CVT-TOKEN that is provided by the Markup Service to the integrator.

When `false`, but the MARKUP_JWT_SECRET environment variable for key exists, then the service will allow integrators to request JWTs and will validate them if they are present.

Default: `false`

Used by feature(s): MarkupService

MARKUP_JWT_SECRET

Description: Sufficiently long random string suitable for generating a key for RS512. This value is passed into the Markup Service environment. If left blank, the Markup Service uses an alternate value as a fallback.

Default: Blank

Used by feature(s): MarkupService

MARKUP_DB_HOST

Description: Host name or IP for the database server that the Markup Service should use. See description of DEFAULT_DB_HOST.

Default: Blank

Used by feature(s): MarkupService

MARKUP_DB_PORT

Description: Port for the database server that the Markup Service should use. See description of DEFAULT_DB_PORT.

Default: Blank

Used by feature(s): MarkupService

MARKUP_DB_NAME

Description: Name of the database that the Markup Service should use. See description of DEFAULT_DB_NAME.

Default: Blank

Used by feature(s): MarkupService

MARKUP_DB_USER

Description: Database user that the Markup Service should use. See description of DEFAULT_DB_USER.

Default: Blank

Used by feature(s): MarkupService

MARKUP_DB_PWD

Description: Database password that the Markup Service should use. See description of DEFAULT_DB_PWD.

Default: Blank

Used by feature(s): MarkupService

MARKUP_DB_USE_SSL

Description: Set to `true` if the database server is configured to use SSL, `false` otherwise. See description of DEFAULT_DB_USE_SSL.

Default: Blank

Used by feature(s): MarkupService

MARKUP_DB_MAX_POOL_SIZE

Description: Maximum connections that the Markup Service will make to the database server.

Default: `10`

Used by feature(s): MarkupService

MARKUP_DB_PROVIDER

Description: Name of the database server that the Markup Service should use. See description of DEFAULT_DB_PROVIDER.

Default: Blank

Used by feature(s): MarkupService

PUBLISHER_SERVICE_HOST

Description: Host name or IP for the Publisher Service.

Default: Blank

Used by feature(s): PublisherService, SearchService

PUBLISHER_AUTH_CLIENT_ID

Description: OAuth clientId from OTDS the Publisher Service should use. Ignored if OTDS_ADMIN_USER was provided. Only needs to be provided if a pre-existing client is being configured.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_AUTH_CLIENT_SECRET

Description: OAuth client secret from OTDS the Publisher Service should use. Ignored if OTDS_ADMIN_USER was provided. Only needs to be provided if a pre-existing client is being configured.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_TRUSTED_SOURCE_ORIGINS

Description: A comma delimited set of origins where source files to be converted are allowed to be retrieved from. Origins are the protocol, fully qualified domain name or IP address and an optional port, with no path. (For example: `http://my.special.server.net/`, `https://www.server.com:1367`). These origins will be presented with an access token to authenticate.

- If neither *`trustedSourceOrigins`* nor *`trustedSourceOriginsAnonymous`* is set, then there is no origin restriction and IV attempts to retrieve source files from the requested file URL.
- If the file URL is at an http endpoint (as opposed to https), no authorization header is sent with the request.
- If either property is defined, then retrievals are restricted to the defined origins.
- Origins defined in *`trustedSourceOrigins`* will pass along an authorization header, whereas origins defined in *`trustedSourceOriginsAnonymous`* won't include an authorization header. The same origin should not be listed in both properties.
- If your Content Server instance is accessible to the IV services at an http endpoint, add its URL (such as `http://ContentServer.internal.net:8080`) to this property and to PUBLICATION_TRUSTED_SOURCE_ORIGINS.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_TRUSTED_SOURCE_ORIGINS_ANONYMOUS

Description: Contains a comma delimited set of origins where no authorization header is sent. These origins will not be presented with an access token during source resolution. See PUBLISHER_TRUSTED_SOURCE_ORIGINS for further description.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_HOST

Description: Host name or IP for the database server that the Publisher Service should use. See description of DEFAULT_DB_HOST.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_PORT

Description: Port for the database server that the Publisher Service should use. See description of DEFAULT_DB_PORT.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_NAME

Description: Name of the database that the Publisher Service should use. See description of DEFAULT_DB_NAME.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_USER

Description: Database user that the database server that the Publisher Service should use. See description of DEFAULT_DB_USER.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_PWD

Description: Database password that the Publisher Service should use. See description of DEFAULT_DB_PWD.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_USE_SSL

Description: Set to `true` if the database server is configured to use SSL, `false` otherwise. See description of DEFAULT_DB_USE_SSL.

Default: Blank

Used by feature(s): PublisherService

SSL support note

SSL support is only available for the PostgreSQL database.

PUBLISHER_DB_SSL_MODE

Description: If PUBLISHER_DB_USE_SSL is `false`, set to `disable`. If PUBLISHER_DB_USE_SSL is `true`, can be set to `prefer`, `require`, `verify-ca`, or `verify-full` in ascending order of security.

Default: `disable`

Used by feature(s): PublisherService

> **Important**
>
> The following PUBLISHER_DB_<> settings are required when PUBLISHER_DB_SSL_MODE is set to either `verify-ca` or `verify-full` and DB_PROVIDER is set to PostgreSQL.

PUBLISHER_DB_PROVIDER

Description: Name of the database server that the Publisher Service should use. See DEFAULT_DB_PROVIDER.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_KEYSTORE_TYPE

Description: Type of key used in the database keystore. Can be `JKS` or `PKCS12`. Mandatory while using `verify-ca` or `verify-full` SSL mode.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_TRUSTSTORE_PATH

Description: Path to truststore file to use with the database. Mandatory while using `verify-ca` or `verify-full` SSL mode.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_TRUSTSTORE_PASSWORD

Description: Password for the truststore file provided in PUBLISHER_DB_TRUSTSTORE_PATH. Mandatory if the truststore is secured with any password.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_TRUSTSTORE_ALIAS

Description: Alias assigned to the certificate or key entry within your truststore. Add only if your trustrore has an alias.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_TRUSTSTORE_ALIAS_PASSWORD

Description: Password associated with the specified alias in your truststore Add only if your truststore has a password secured alias.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_KEYSTORE_PATH
Description: This optional setting is the path to the keystore file to use with the database.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_KEYSTORE_PASSWORD
Description: Password for the keystore file provided in PUBLISHER_DB_KEYSTORE_PATH.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_KEYSTORE_ALIAS
Description: Alias assigned to the certificate or key entry within your keystore. Add only if your keystore has an alias.

Default: Blank

Used by feature(s): PublisherService

PUBLISHER_DB_KEYSTORE_ALIAS_PASSWORD
Description: Password associated with the specified alias in your keystore Add only if your keystore has a password secured alias.

Default: Blank

Used by feature(s): PublisherService

PUBLICATION_SERVICE_HOST
Description: Host name or IP for the Publication Service. See description of DEFAULT_HOST.

Default: Blank

Used by feature(s): AssetService, PublicationService

PUBLICATION_SERVICE_PORT
Description: Port number to use for the Publication Service. See description of DEFAULT_PORT.

Default: `3356`

Used by feature(s): AssetService, PublicationService

PUBLICATION_AUTH_CLIENT_ID
Description: OAuth clientId from OTDS the Publication Service should use.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_AUTH_CLIENT_SECRET
Description: OAuth client secret from OTDS the Publication Service should use.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_TRUSTED_SOURCE_ORIGINS

Description: A comma delimited set of origins where source files to be converted are allowed to be retrieved from. Origins are the protocol, fully qualified domain name or IP address and an optional port, with no path. (For example: `http://my.special.server.net/`, `https://www.server.com:1367`). These origins will be presented with an access token to authenticate.

- If neither *`trustedSourceOrigins`* nor *`trustedSourceOriginsAnonymous`* is set, then there is no origin restriction and IV attempts to retrieve source files from the requested file URL.
- If the file URL is at an http endpoint (as opposed to https), no authorization header is sent with the request.
- If either property is defined, then retrievals are restricted to the defined origins.
- Origins defined in *`trustedSourceOrigins`* will pass along an authorization header, whereas origins defined in *`trustedSourceOriginsAnonymous`* won't include an authorization header. The same origin should not be listed in both properties.
- If your Content Server instance is accessible to the IV services at an http endpoint, add its URL (such as `http://ContentServer.internal.net:8080`) to this property and to PUBLISHER_TRUSTED_SOURCE_ORIGINS.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_TRUSTED_SOURCE_ORIGINS_ANONYMOUS

Description: Contains a comma delimited set of origins where no authorization header is sent. These origins will not be presented with an access token during source resolution. See PUBLICATION_TRUSTED_SOURCE_ORIGINS for further description.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_HOST

Description: Host name or IP for the database server that the Publication Service should use. See description of DEFAULT_DB_HOST.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_PORT

Description: Port for the database server that the Publication Service should use. See description of DEFAULT_DB_PORT.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_NAME

Description: Name of the database that the Publication Service should use. See description of DEFAULT_DB_NAME.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_USER

Description: Database user that the database server that the Publication Service should use. See description of DEFAULT_DB_USER.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_PWD

Description: Database password that the Publication Service should use. See description of DEFAULT_DB_PWD.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_USE_SSL

Description: Set to `true` if the database server is configured to use SSL, `false` otherwise. See description of DEFAULT_DB_USE_SSL

Default: Blank

Used by feature(s): PublicationService

SSL support note

SSL support is only available for the PostgreSQL database.

PUBLICATION_DB_SSL_MODE

Description: If PUBLCATION_DB_USE_SSL is `false`, set to `disable`. If PUBLICATION_DB_USE_SSL is `true`, can be set to `prefer`, `require`, `verify-ca`, or `verify-full` in ascending order of security.

Default: `disable`

Used by feature(s): PublicationService

Important

The following PUBLICATION_DB_<> settings are required when PUBLICATION_DB_SSL_MODE is set to either `verify-ca` or `verify-full` and DB_PROVIDER is set to PostgreSQL.

PUBLICATION_DB_PROVIDER

Description: Name of the database server that the Publication Service should use. See DEFAULT_DB_PROVIDER.

Default: `disable`

Used by feature(s): PublicationService

PUBLICATION_DB_KEYSTORE_TYPE

Description: Type of key used in the database keystore. Can be `JKS` or `PKCS12`. Mandatory while using `verify-ca` or `verify-full` SSL mode.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_TRUSTSTORE_PATH

Description: Path to truststore file to use with the database. Mandatory while using `verify-ca` or `verify-full` SSL mode.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_TRUSTSTORE_PASSWORD

Description: Password for the truststore file provided in PUBLICATION_DB_TRUSTSTORE_PATH. Mandatory if the truststore is secured with any password.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_TRUSTSTORE_ALIAS

Description: Alias assigned to the certificate or key entry within your truststore. Add only if your trustrore has an alias.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_TRUSTSTORE_ALIAS_PASSWORD

Description: Password associated with the specified alias in your truststore Add only if your truststore has a password secured alias.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_KEYSTORE_PATH

Description: This optional setting is the path to the keystore file to use with the database.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_KEYSTORE_PASSWORD

Description: Password for the keystore file provided in PUBLICATION_DB_KEYSTORE_PATH.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_KEYSTORE_ALIAS

Description: Alias assigned to the certificate or key entry within your keystore. Add only if your keystore has an alias.

Default: Blank

Used by feature(s): PublicationService

PUBLICATION_DB_KEYSTORE_ALIAS_PASSWORD

Description: Password associated with the specified alias in your keystore Add only if your keystore has a password secured alias.

Default: Blank

Used by feature(s): PublicationService

ACCESS_CHECK_TIMEOUT_SECONDS

Description: Sets the Publication Service access check timeout in seconds.

Default: `3`

Used by feature(s): PublicationService

SEARCH_SERVICE_HOST

Description: Host name or IP for the Search Service. See description of DEFAULT_HOST.

Default: Blank

Used by feature(s): SearchService

SEARCH_SERVICE_PORT

Description: Port number to use for the Search Service. See description of DEFAULT_PORT.

Default: `3357`

Used by feature(s): SearchService

SEARCH_AUTH_CLIENT_ID

Description: OAuth clientId from OTDS that the Search Service should use.

Default: Blank

Used by feature(s): SearchService

SEARCH_AUTH_CLIENT_SECRET

Description: OAuth client secret from OTDS that the Search Service should use.

Default: Blank

Used by feature(s): SearchService

VIEWER_HOST

Description: Host name or IP for the Viewer Service. See description of DEFAULT_HOST.

Default: Blank

Used by feature(s): ViewerService

VIEWER_PORT

Description: Port number to use for the Viewer Service. See description of DEFAULT_PORT.

Default: `3358`

Used by feature(s): ViewerService

OTCS_ADMIN_USER
Description: Content Server admin user. Should only be provided if installing IV for Content Server.

Default: Blank

Used by feature(s): Content Server

OTCS_ADMIN_PW
Description: Content Server admin password. Should only be provided if installing IV for Content Server.

Default: Blank

Used by feature(s): Content Server

OTCS_ADMIN_ORIGIN
Description: Content Server URL. Only used if OTCS_ADMIN_USER is supplied.

Default: Default value is `http://<host>:8080/OTCS/cs`, which is invalid. This value must be customized if used.

Used by feature(s): Content Server

OTCS_RESOURCE_NAME
Description: Content Server's resource name in OTDS. Only used if OTCS_ADMIN_USER is supplied.

Default: `cs`

Used by feature(s): Content Server

4.5 Verify your installation

When the setup of your environment is complete, verify that your Intelligent Viewing installation is functional.

Note: Any folder, whether created manually or by the installer, that the Intelligent Viewing services need to use must have the correct access permissions granted to the users accounts running those services.

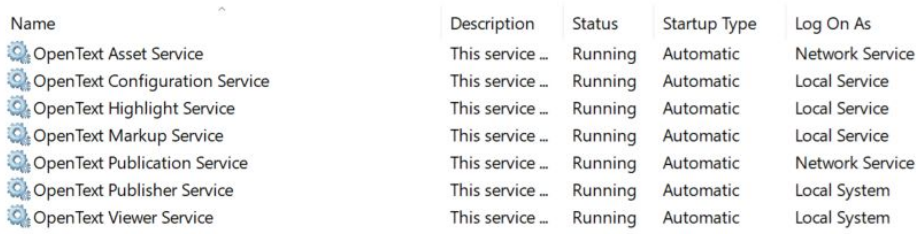
If you installed Intelligent Viewing for Content Server, you can verify your installation by attempting to view a document in Content Server.

Note: The Metis Service is removed in Intelligent Viewing beginning with version 23.3 and the demo webapp can no longer be used to verify your installation.

You can manually check the installation log file and also individual installation items by visiting the following pages:

Installation checklist

- Check your Windows **Services** panel and verify that all of the installed services are running.

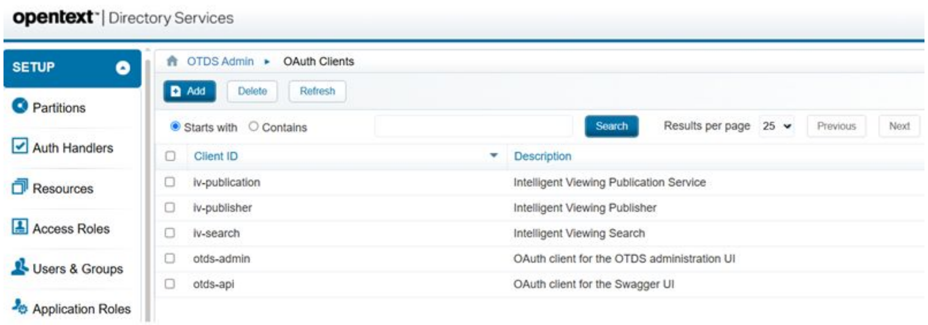


Name	Description	Status	Startup Type	Log On As
OpenText Asset Service	This service ...	Running	Automatic	Network Service
OpenText Configuration Service	This service ...	Running	Automatic	Local Service
OpenText Highlight Service	This service ...	Running	Automatic	Local Service
OpenText Markup Service	This service ...	Running	Automatic	Local Service
OpenText Publication Service	This service ...	Running	Automatic	Network Service
OpenText Publisher Service	This service ...	Running	Automatic	Local System
OpenText Viewer Service	This service ...	Running	Automatic	Local System

Note: The **OpenText Highlight Service** refers to the **Search Service** that was selected during installation.

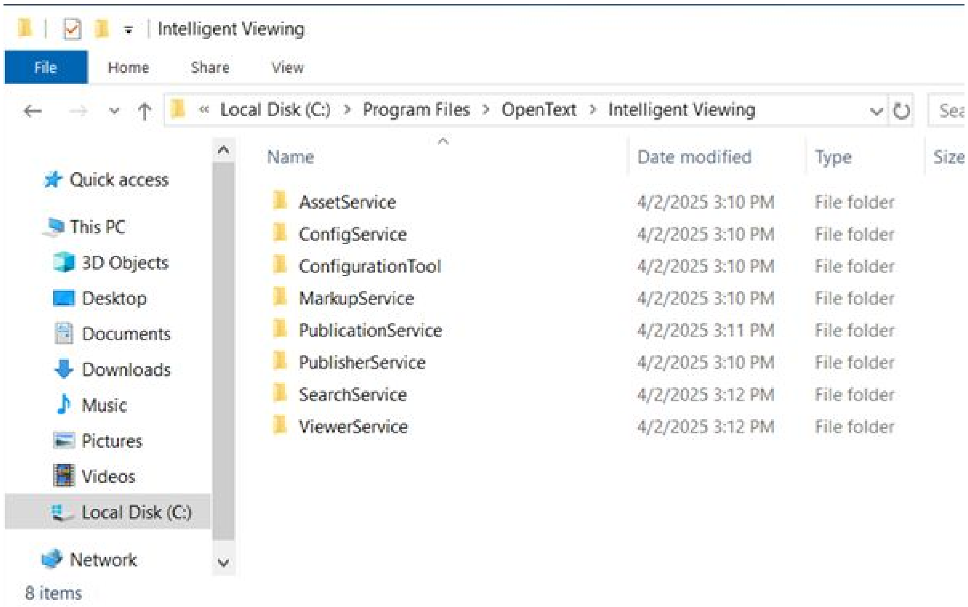
If you are running services under a custom account, see "Running Services under a custom account" on page 63.

- In OTDS, verify the following additions:
  - From the **OTDS Admin > OAuth Clients** page, verify that the Intelligent Viewing client IDs are added.



□	Client ID	Description
□	iv-publication	Intelligent Viewing Publication Service
□	iv-publisher	Intelligent Viewing Publisher
□	iv-search	Intelligent Viewing Search
□	otds-admin	OAuth client for the OTDS administration UI
□	otds-api	OAuth client for the Swagger UI

  - From the **OTDS Admin > License Keys** page, verify that the `Viewing` license key is added.
  - From the **OTDS Admin > Resources** page, verify that the `iv` resource is added.
- Verify the presence of the service install folders located in `C:\Program Files\OpenText\Intelligent Viewing` by default.



Name	Date modified	Type	Size
AssetService	4/2/2025 3:10 PM	File folder	
ConfigService	4/2/2025 3:10 PM	File folder	
ConfigurationTool	4/2/2025 3:10 PM	File folder	
MarkupService	4/2/2025 3:10 PM	File folder	
PublicationService	4/2/2025 3:11 PM	File folder	
PublisherService	4/2/2025 3:10 PM	File folder	
SearchService	4/2/2025 3:12 PM	File folder	
ViewerService	4/2/2025 3:12 PM	File folder	

4.6 Running Services under a custom account

If your environment requires Intelligent Viewing services to run under a custom account, follow these important guidelines before changing the account the service is running as:

- Stop the service before making any changes to the account the service is running under.
- Delete all log files in the corresponding log folder (after taking backup copies, if required) but do not delete the folder itself. Logs created by a different account might not be writable by the new account, which can cause service failures.
- Grant the account full access to the path provided in the ARTIFACT_VOLUMES_ROOT install property. The Publication, Publisher, and Asset services must have full access permissions on the artifact cache folder to ensure they can properly read from and write to the cached artifacts.
- Grant the `Log on as a service` right to the custom account. This right allows the user account to be used to start Windows services.
  - When you configure the account through the Windows Services management console (`services.msc`), this right is usually granted automatically.
  - For detailed guidance on granting the `Log on as a service` right, see the official Microsoft documentation: Log on as a service - Microsoft Learn (https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/log-on-as-a-service)

4.7 Intelligent Viewing Sample Application (IVSA)

A sample application that provides a basic interface to Intelligent Viewing functionality is externally available. For download, setup, and configuration instructions for the Intelligent Viewing Sample Application (IVSA), refer to:

https://github.com/opentext/ivsa

4.8 (Optional) Use Microsoft Office to process files

Intelligent Viewing can optionally use a copy of Microsoft Office 365 or Microsoft Visio installed on the server to convert Microsoft formats. This type of conversion is facilitated by a component named CSF Writer that interfaces between the Publisher Service and the Microsoft software. This component is installed using an MSI that is created in the PublisherService directory when the Publisher Service is installed. Processing Office or Visio files with CSF Writer uses a loader named Bi2dl instead of the default loader Lo2dl for these formats. Bi2dl then delegates the conversation to CSF Writer, which in turn delegates to Office or Visio.

Files processed in this way appear similar to how they would appear in office but may be processed more slowly. Additionally, files cannot be processed in parallel. As a result, the throughput is slower overall, especially if there is heavy load on the server. The CSFWriter installer is provided under the Intelligent Viewing Publisher Service folder at `C:\Program Files\OpenText\Intelligent Viewing\PublisherService`. After installation the CSF Writer program files are located in its installation directory `C:\Program Files\OpenText\CSF Writer`.

CSF Writer requirements

- Install Intelligent Viewing as described in "Run the Intelligent Viewing installer" on page 38.
- Install and activate a version of Microsoft Office 365, including Microsoft Visio Professional. This is required to render Office formats (DOC, DOCX, XLS, XLSX, PPT, and PPTX) as well as Visio files.
- A user account with administrator privileges and `Log on as a service` rights is required to run the Publisher Service.

To install CSF Writer

1. Run the CSF Writer .MSI file. This file is named `CSFWriter_<version>.msi`, and is created as part of the IV Installation process.
2. Accept the license agreement when prompted.
3. Click **Install**. Once the installation is finished, click **Finish**.

To enable CSF Writer

1. Stop the OpenText Publisher Service.

2. Configure OpenText Publisher Service to run under a user account. This account requires both administrator privileges and `Log On As a Service` rights.
3. In the services console, right-click **OpenText Publisher Service**.
4. Click **Properties**, and then select the **Log On** tab.
5. Select **This account** and enter the user credentials.
6. Associate file extensions with the Bi2dl loader:

   a. Open a command prompt window.

   b. Navigate to `<Intelligent Viewing Installation Directory>\PublisherService\bin`.

   c. For each Office or Visio file extension, run the following command:

   ```
   loaderconfig --extensionset bi2dl <extension>
   ```

   d. Verify the configuration by running the following command:

   ```
   loaderconfig --extensionlist bi2dl
   ```

   e. Confirm all required extensions appear in the list.
7. Restart the Publisher Service.

Chapter 5

Updating the IV service configuration

After installation completes, the environmental variables that establish the configuration for the various Intelligent Viewing services are stored in the following files:

Service	ENV config file path
Asset Service	<INSTALLDIR>/AssetService/config/env.conf
Config Service	<INSTALLDIR>/ConfigService/config/env.conf
Markup Service	<INSTALLDIR>/MarkupService/.env
Publication Service	<INSTALLDIR>/PublicationService/config/env.conf
Publisher Service	<INSTALLDIR>/PublisherService/config/env.conf
Search Service	<INSTALLDIR>/SearchService/.env
Viewer Service	<INSTALLDIR>/ViewerService/.env

If you have installed the Configuration Tool, you can quickly modify these service configuration files at any time, restart the services, and download log files through the tool. See “Using the Configuration Tool“ on page 69. Parameter details for each service are listed under Common Properties.

If you have manually updated these .env files, restart the Windows service(s) corresponding to the IV service(s) for which configuration has changed.

The default value for <INSTALLDIR> is `C:\Program Files\ OpenText\ Intelligent Viewing` as described in “Installer properties” on page 40. This section also serves as a guide for the meaning of many of the environment variables. See “Error logging“ on page 105 for how to set environment variables required for logging.

Chapter 6

Using the Configuration Tool

When the optional **Configuration Tool** feature is installed (see "Run the Intelligent Viewing installer" on page 38 for Windows or, beginning with version 25.4, for Linux *OpenText Intelligent Viewing - Linux Install Guide (CLIVSA-IGL)*), a user interface is provided for Administrators to complete or update the Intelligent Viewing configuration after installation.

Using this tool, you can configure security details, database settings, RabbitMQ settings, and the Viewing and Transformation Service settings for each installed service. You can also start and stop these services with the tool and quickly download service log files.

Any changes that are made to the settings provided in the forms of this tool are written to the environment configuration files that would otherwise require a manual update. The tool will validate your changes and provide you with assurance that your configuration and restart of the services is successful.

The main purpose of this tool is to aid in a successful configuration and deployment of Intelligent Viewing, and to minimize and troubleshoot any potential issues.

The basic steps needed to update your configuration after installation are:

1. Run the Configuration Tool application.
2. Obtain and enter an Authorization token in the **Authorization** page.
3. Update the individual service settings as needed.
4. Update the individual service settings for the security, database, and RabbitMQ settings as needed.
5. Restart any services that have been updated and ensure all installed services are running.
6. View and Download service logs if needed.

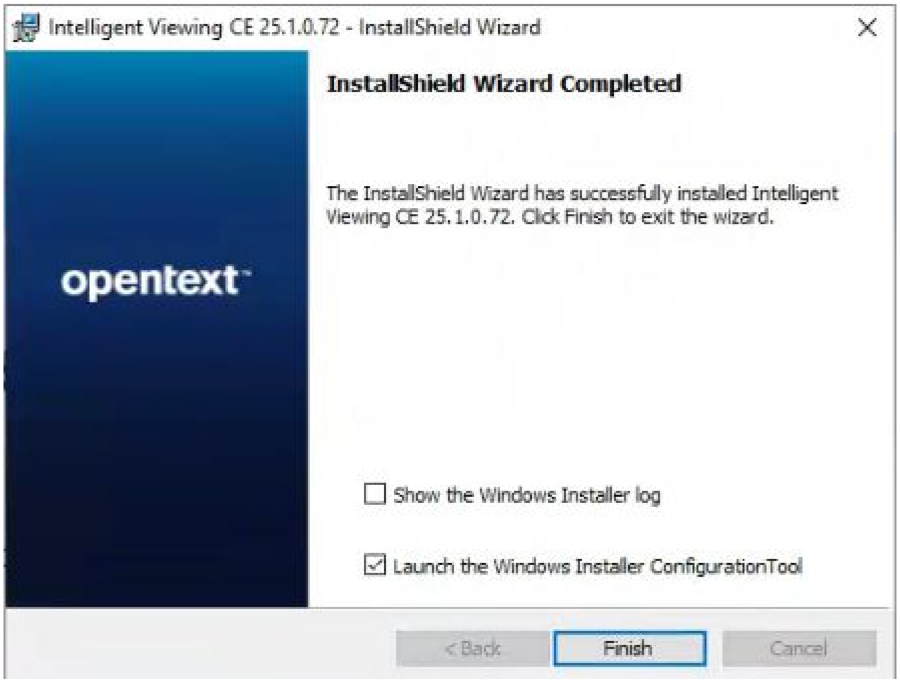
6.1 Launch the config-tool application

When the optional **Configuration Tool** feature is installed (see "Run the Intelligent Viewing installer" on page 38 for Windows or *OpenText Intelligent Viewing - Linux Install Guide (CLIVSA-IGL)* for Linux), a user interface is provided for configuring Intelligent Viewing after installation.

To run the Configuration Tool:

1. **On Windows** – Do one of:

   - **During installation**: As part of the Intelligent Viewing installation process, an option is provided on the **Finish** screen to **Launch the Windows Installer Configuration Tool**.



Intelligent Viewing CE 25.1.0.72 - InstallShield Wizard

InstallShield Wizard Completed

The InstallShield Wizard has successfully installed Intelligent Viewing CE 25.1.0.72. Click Finish to exit the wizard.

□ Show the Windows Installer log

☑ Launch the Windows Installer ConfigurationTool

   - **Through the shortcut**: During the Intelligent Viewing installation, a shortcut to launch the configuration tool is automatically created on the desktop. You can double-click the config-tool shortcut to launch the tool.
   - **Manually running the config-tool**: You can navigate to the `INSTALLDIR\ConfigurationTool` folder and run the `config-tool.exe` file.

2. **On Linux**:

   - **Manually run the config-tool**: Navigate to `[INSTALL_DIR]/ConfigurationTool` and run the `config-tool.sh` file.

3. When the tool is started, an **Authorization** page appears in which you must enter a valid authorization token before you will be able to use the tool. To obtain a token, see "Get an authorization token" on page 71.

6.1.1 Get an authorization token

When you launch the Configuration Tool interface, you must first enter a valid authorization token in the resulting **Authorization** page to use the tool.

To get a token:

1. **On Windows** – You can obtain an authorization token through one of these three methods:

    - **During installation**: As part of the Intelligent Viewing installation process, a **Launch the Windows Installer Configuration Tool** option is provided on the final screen. If you select this check box and click **Finish**, a command prompt is launched in which you can view the token.
    - **Through the shortcut**: During the installation, a shortcut to launch the configuration tool is automatically created on the desktop. You can double-click the shortcut to launch the command prompt, where the token will be displayed.
    - **Manually running the config-tool**: You can navigate to the `INSTALLDIR\ConfigurationTool` folder and run the `config-tool.bat` file. This will open the command prompt, and the token will be displayed.

2. **On Linux** – Manually run the config-tool by navigating to `INSTALLDIR\ConfigurationTool` and run the `config-tool.sh` file. This will open the command prompt, where the token will be displayed.

3. Keeping the command prompt window open, **Copy** the token received and **Paste** it into the Authorization page **Enter token** field, then click **Connect**.

4. When the token is correct and validated, the **Service Configuration** page is displayed, allowing you to continue the configuration.

> **!** **Important**
>
> Do not close the command prompt window while using the Configuration Tool. Doing so will end your session with the tool.

6.2 Configure service settings

The**Service Configuration** tab is used to configure the environment variables for each of the installed services.

To configure individual service properties:

1. On the **Service Configuration** page, click one of the installed services listed on the left panel.
2. Edit the desired properties in the form that displays on the right panel.
3. Click **Save**.

4. Click the **Start Service** or **Restart Service** button for any services that have been updated.

To configure all service properties:

1. On the **Service Configuration** page, click one of the installed services listed on the left panel.
2. Edit the desired properties in the form that displays on the right panel.
3. Click **Save to all**. This button is visible only when more than one service on the machine is using the same value for the updated setting.
4. Click **Confirm** if you want to apply your changes to all affected services.

   ! **Important**

   If only one value is updated, only that value is updated for the corresponding services. All remaining values are not affected.

5. Upon success, you are prompted to click the **Start/Restart** button to start/restart all of the services that have been updated. If you choose not to restart all services at this time, you will need to start/restart each service manually.

- The available services are listed on the left panel and when each service is clicked, the service corresponding form fields are displayed and can be scrolled in the right panel. The form for the first available service displays by default until another service is selected.
- The initial values displayed are pulled from the .env file of the selected service. For some values, if the .env file value is empty, the default value from JSON will be used.
- The Configuration Tool honors encryption and will display encrypted property values as asterisks (****). If you want to update an encrypted parameter value, enter the new value in the **Service Configuration** form input field and click **Save**. The encrypted version of this updated value is written to the associated configuration file and the service will use this value the next time it is started or restarted.
- For some properties, dependent properties do not appear unless that value is enabled.
- The default **Security**, **Database**, and **RabbitMQ** properties that will be used for each of the installed services are maintained in the environment configuration file. If individual services require different settings, these settings can be overridden for each service by clicking the **Edit** button provided beneath the **Security Configuration**,**Database Configuration**, or **RabbitMQ Configuration** option. When clicked, a dialog is presented that is auto-populated with the env file property values that can be updated individually for each service.
- Any changes that are made in the Configuration Tool form fields will activate the **Save** and **Save to all** buttons. When clicked, a check is made to verify that all mandatory fields are filled. If not, a message displays and the form(s) are not

saved until all required fields are filled. When the forms can be saved and the configuration file values are updated, a Success or Failure message displays indicating the status of the configuration. Note that clicking **Save** only saves the values in the selected service configuration file and does not start or restart the service. When using the **Save to all** button to save changes to all affected services, you will be prompted to **Start/Restart** all updated services. See "Start and stop services" on page 95.

- If you have made changes and click on another service without saving those changes, you will be prompted to either **Leave** the page without saving changes (changes are discarded), or **Cancel** the prompt so you can save your changes in the currently selected service. This prompt will also display when you attempt a browser refresh without having saved your changes.
- Click the **Revert** button if you want to fetch the latest saved data from the selected service configuration file.

6.2.1 Viewer Service settings

The Viewing Services include the Viewer Service, Markup Service, and Highlight Service. This page controls the settings in the Viewer Service configuration file that is installed by default to `<INSTALLDIR>/ViewerService/.env`.

Port

Parameter name: PORT

Default value: 3358

Required: YES

Type: Integer

Description: Viewer Service Port number.

Document Root

Parameter name: DOC_ROOT

Default value: http://developer.opentext.com

Required: NO

Type: URL

Description: Root URL to the devx portal where REST clients can go for more information about the API (used in HAL LINKS).

Viewer Service Origin URL

Parameter name: VIEWER_AUTHORITY

Default value: not set

Required: YES

Type: URL

Description: Origin URL used for the Viewer Service (for example, http://<IP_HOST>:<PORT>.com).

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

For **Security Configuration** settings details, see "Security settings" on page 84. You can click the **Edit** button to display a dialog in which you can modify the Security settings available for this service. Saved changes will override the default values that are set in the environment configuration file for this service only.

6.2.2 Highlight Service settings

The Viewing Services include the Viewer Service, Markup Service, and Highlight Service. This page controls the settings in the Highlight Service configuration file that is installed by default to `<INSTALLDIR>/SearchService/.env`.

Port

Parameter name: PORT

Default value: 3357

Required: YES

Type: Integer

Description: Search Service Port number.

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

Auth Client Id

Parameter name: AUTH_CLIENT_ID

Default value: not set

Required: YES

Type: Encrypted string

Description: The OAuth clientId from OTDS that the service should use. Value is dependent on RESOURCE value and will change when resource changes.

Auth Client Secret

Parameter name: AUTH_CLIENT_SECRET

Default value: not set

Required: YES

Type: Encrypted string

Description: The OAuth client secret from OTDS that the service should use. Value is dependent on RESOURCE value and will change when resource changes.

Publication Service Origin URL

Parameter name: PUBLICATION_AUTHORITY

Default value: not set

Required: YES

Type: URL

Description: Origin URL to the Publication Service (for example, http://<IP_HOST>:<PORT>).

Search Service Origin URL

Parameter name: SEARCH_AUTHORITY

Default value: not set

Required: YES

Type: URL

Description: Origin URL to the Search Service (for example, http://<IP_HOST>:<PORT>).

For **Security Configuration** settings details, see "Security settings" on page 84. Clicking the **Edit** button displays a dialog in which you can modify the Security settings available for this service. Saved changes will override the default values that are set in the environment configuration file for this service only.

6.2.3 Markup Service settings

The Viewing Services include the Viewer Service, Markup Service, and Highlight Service. This page controls the settings in the Markup Service configuration file that is installed by default to `<INSTALLDIR>/MarkupService/.env`.

Port

Parameter name: PORT

Default value: 3352

Required: YES

Type: Integer

Description: Markup Service Port number.

Markup Service Origin URL

Parameter name: MARKUP_AUTHORITY

Default value: not set

Required: YES

Type: URL

Description: Origin URL to the Markup Service (for example, http://<IP_HOST>:<PORT>.com).

Node Environment

Parameter name: NODE_ENV

Default value: production

Required: YES

Type: Choice: **production/development**

Description: Run as either **development** or **production**. If **Development**, the GraphQL query playground is hosted at `[host]://markup/api/v1/graphql`, otherwise, the playground is not available.

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

Enable Role Based Access Control

Parameter name: ENABLE_ROLE_BASED_ACCESS_CONTROL

Default value: False

Required: YES when the HTTPS protocol is enabled

Type: Boolean

Description: Enables or disables role based access control. The Markup Service must apply finer grained access control to markups for individual users.

When **True**, then the Markup Service requires an additional header X-CVT-TOKEN that is provided by the Markup Service to the integrator.

Markup JWT Secret

Parameter name: MARKUP_JWT_SECRET

Default value: not set

Required: YES when the HTTPS protocol is enabled

Type: Encrypted string

Description: If ENABLE_ROLE_BASED_ACCESS_CONTROL=true OR header X-CVT-TOKEN was set, then each Markup Service must also have the MARKUP_JWT_SECRET environment variable set to a secret.

For **Security Configuration** and **Database Configuration** settings details, see "Security settings" on page 84. and "Database settings" on page 87. Clicking the **Edit** button displays a dialog in which you can modify the Security and Database settings available for this service. Saved changes will override the default values that are set in the environment configuration file for this service only.

6.2.4 Configuration Service settings

The Transformation Services include the Configuration Service, Publisher Service, Publication Service, and Asset Service. This page controls the settings in the Configuration Service configuration file that is installed by default to `<INSTALLDIR>/ConfigService/config/env.conf`.

Configuration Service Origin URL

Parameter name: AUTHORITY

Default value: not set

Required: YES

Type: URL

Description: Origin URL to Configuration Service (for example, http://<IP_HOST>:<PORT>.com)

Port

Parameter name: PORT

Default value: 3351

Required: YES

Type: Integer

Description: Configuration Service Port number.

Internal Rest Port

Parameter name: INTERNAL_REST_PORT

Default value: 3351

Required: YES

Type: Integer

Description: Configuration Service Internal Rest Port number. Updates automatically when PORT is updated.

Peer Rest Port

Parameter name: PEER_REST_PORT

Default value: 3351

Required: YES

Type: Integer

Description: Configuration Service Peer Rest Port number. Updates automatically when PORT is updated.

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

For **Security Configuration** and **Database Configuration** settings details, see "Security settings" on page 84. and "Database settings" on page 87. Clicking the **Edit** button displays a dialog in which you can modify the Security and Database settings available for this service. Saved changes will override the default values that are set in the environment configuration file for this service only.

6.2.5 Publisher Service settings

The Transformation Services include the Configuration Service, Publisher Service, Publication Service, and Asset Service. This page controls the settings in the Publisher Service configuration file that is installed by default to `<INSTALLDIR>/PublisherService/config/env.conf`.

Markup Service Origin URL

Parameter name: MARKUP_SERVICE_ORIGIN

Default value: not set

Required: YES

Type: URL

Description: Origin URL to Markup Service (for example, http://<IP_HOST>:<PORT>.com).

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

Auth Client Id

Parameter name: AUTH_CLIENT_ID

Default value: not set

Required: YES

Type: Encrypted string

Description: OAuth clientId from OTDS the service should use. Value is dependent on RESOURCE value and will change when resource changes.

Auth Client Secret

Parameter name: AUTH_CLIENT_SECRET

Default value: not set

Required: YES

Type: Encrypted string

Description: OAuth client secret from OTDS the service should use. Value is dependent on RESOURCE value and will change when resource changes.

Trusted Source Origins

Parameter name: TRUSTED_SOURCE_ORIGINS

Default value: not set

Required: NO

Type: Comma separated hostnames

Description: Comma separated list of origins that source files are permitted to be accessed from. Origins are the protocol, fully qualified domain name or IP address and an optional port, with no path. For example, `http://my.special.server.net/`.

Trusted Source Origins Anonymous

Parameter name: TRUSTED_SOURCE_ORIGINS_ANONYMOUS

Default value: not set

Required: NO

Type: Comma separated hostnames

Description: Comma separated list of origins that source files are permitted to be accessed from without any authentication.

Asset Service Artifacts URL

Parameter name: MKONDO_BLOB_BASE_URL

Default value: not set

Required: YES

Type: URL

Description: URL for accessing the artifacts.

Artifact Storage Path

Parameter name: MKONDO_BLOB_ROOTS

Default value: not set

Required: YES

Type: Filepath

Description: Path to a local or share folder where artifacts should be stored.

For **Security Configuration**, **Database Configuration**, and **RabbitMQ Configuration** settings details, see "Security settings" on page 84, "Database settings" on page 87, and "RabbitMQ settings" on page 93.. Clicking the **Edit** button displays a dialog in which you can modify the Security, Database, and RabbitMQ settings available for this service. Saved changes will override the default values that are set in the environment configuration file for this service only.

6.2.6 Publication Service settings

The Transformation Services include the Configuration Service, Publisher Service, Publication Service, and Asset Service. This page controls the settings in the Publication Service configuration file that is installed by default to `<INSTALLDIR>/PublicationService/config/env.conf`.

Publication Service Origin URL

Parameter name: AUTHORITY

Default value: not set

Required: YES

Type: URL

Description: Origin URL to Publication Service (for example, http://<IP_HOST>:<PORT>.com).

Port

Parameter name: PORT

Default value: 3356

Required: YES

Type: Integer

Description: Publication Service Port number.

Internal Rest Port

Parameter name: INTERNAL_REST_PORT

Default value: 3356

Required: YES

Type: Integer

Description: Publication Service Internal Rest Port number.

Peer Rest Port

Parameter name: PEER_REST_PORT

Default value: 3356

Required: YES

Type: Integer

Description: Publication Service Peer Rest Port number.

Auth Client Id

Parameter name: AUTH_CLIENT_ID

Default value: not set

Required: YES

Type: Encrypted string

Description: OAuth clientId from OTDS the service should use. Value is dependent on RESOURCE value and will change when resource changes.

Auth Client Secret

Parameter name: AUTH_CLIENT_SECRET

Default value: not set

Required: YES

Type: Encrypted string

Description: OAuth client secret from OTDS the service should use. Value is dependent on RESOURCE value and will change when resource changes.

Trusted Source Origins

Parameter name: TRUSTED_SOURCE_ORIGINS

Default value: not set

Required: NO

Type: Comma separated hostnames

Description: Comma separated list of origins that source files are permitted to be accessed from. Origins are the protocol, fully qualified domain name or IP address and an optional port, with no path. For example, `http://my.special.server.net/`.

Trusted Source Origins Anonymous

Parameter name: TRUSTED_SOURCE_ORIGINS_ANONYMOUS

Default value: not set

Required: NO

Type: Comma separated hostnames

Description: Comma separated list of origins that source files are permitted to be accessed from without any authentication.

Asset Service Artifacts URL

Parameter name: MKONDO_BLOB_BASE_URL

Default value: not set

Required: YES

Type: URL

Description: Hostname to embed in HAL links. Must be set properly for http or https depending on the protocol being used.

Artifact Storage Path

Parameter name: MKONDO_BLOB_ROOTS

Default value: not set

Required: YES

Type: Filepath

Description: Path to a local or share folder where artifacts should be stored.

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

Access Check Timeout Seconds

Parameter name: ACESS_CHECK_TIMEOUT_SECONDS

Default value: 3

Required: NO

Type: Integer

Description: Value (in seconds) to set the Publication Service access check timeout.

For **Security Configuration**, **Database Configuration**, and **RabbitMQ Configuration** settings details, see "Security settings" on page 84, "Database settings" on page 87, and "RabbitMQ settings" on page 93. Clicking the **Edit** button displays a dialog in which you can modify the Security, Database, and RabbitMQ settings available for this service. Saved changes will override the default values that are set in the environment configuration file for this service only.

6.2.7 Asset Service settings

The Transformation Services include the Configuration Service, Publisher Service, Publication Service, and Asset Service. This page controls the settings in the Asset Service configuration file that is installed by default to `<INSTALLDIR>/AssetService/config/env.conf`.

Publication Service Origin URL

Parameter name: PUBLICATION_SERVICE_ORIGIN

Default value: not set

Required: YES

Type: URL

Description: Origin URL to Publication Service (for example, http://<IP_HOST>:<PORT>.com).

Port

Parameter name: PORT

Default value: 3350

Required: YES

Type: Integer

Description: Asset Service Port number.

Internal REST Port

Parameter name: INTERNAL_REST_PORT

Default value: 3350

Required: YES

Type: Integer

Description: Asset Service Internal Rest Port number.

Peer REST Port

Parameter name: PEER_REST_PORT

Default value: 3350

Required: YES

Type: Integer

Description: Asset Service Peer Rest Port number.

Artifact Volumes Root

Parameter name: ARTIFACT_VOLUMES_ROOT

Default value: C:/IVArtifacts

Required: YES

Type: Filepath

Description: Path to a local or share folder where artifacts should be stored.

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

For **Security Configuration** settings details, see “Security settings” on page 84. Clicking the **Edit** button displays a dialog in which you can modify the Security settings available for this service. Saved changes will override the default values that are set in the environment configuration file for this service only.

6.2.8 Security settings

You can set overriding Security settings values for each of the individual services through the **Service Configuration** pages **Edit** buttons.

OTDS Origin URL

Parameter name: OTDS_ORIGIN

Default value: not set

Required: YES

Type: URL

Description: Fully qualified `http://<hostname>:<port>` URL of OTDS server where the Intelligent Viewing license is hosted. While this value is read from the .env file, it can be edited.

Resource

Parameter name: RESOURCE

Default value: not set

Required: YES

Type: Encrypted string

Description: A string value that is the base64 encoded result of `<OTDS-resource-id>:<OTDS-resource-secret>` for licensing. While this value is read from the .env file, it can be edited to add a new resource.

Enforce CORS Origins

Parameter name: ENFORCE_CORS_ORIGINS

Default value: False

Required: YES

Type: Boolean

Description: Enable/disable CORS whitelist enforcement.

Note: If the parameter ENFORCE_CORS_ORIGINS is set to `True`, and values exist for any of its dependent parameter fields, those values will be cleared in the selected service env file if you set **Enforce CORS Origins** to **False**. In this case, upon clicking **Save**, a warning message appears to **Confirm** that you want to clear the fields for the following:

- CORS Origins List
- CORS Additional Headers List

CORS Origins List

Parameter name: CORS_ORIGINS_LIST

Default value: not set

Required: YES when ENFORCE_CORS_ORIGINS = true

Type: Comma separated hostnames

Description: A comma separated list of CORS origin strings.

CORS Additional Headers List

Parameter name: CORS_ADDITIONAL_HEADERS_LIST

Default value: not set

Required: YES when ENFORCE_CORS_ORIGINS = true

Type: Comma separated hostnames

Description: A comma separated list of header names to be added to the default list of allowed headers.

Enforce Forwarded Hosts

Parameter name: ENFORCE_FORWARDED_HOSTS

Default value: False

Required: YES

Type: Boolean

Description: Enable/disable Forwarded host whitelist enforcement.

Note: If the parameter ENFORCE_FORWARDED_HOSTS is set to `True`, and values exist for any of its dependent parameter fields, those values will be cleared in the selected service env file if you set **Enforce Forwarded Hosts** to **False**. In this case, upon clicking **Save**, a warning message appears to **Confirm** that you want to clear the fields for the following: **Forwarded Hosts List**

Forwarded Hosts List

Parameter name: FORWARDED_HOSTS_LIST

Default value: not set

Required: YES when ENFORCE_FORWARDED_HOSTS = true

Type: Comma separated hostnames

Description: A comma separated list of Forwarded host strings.

Enable HTTPS

Parameter name: ENABLE_HTTPS

Default value: False

Required: YES

Type: Boolean

Description: Protocol to use for communication to the various hosts. If set to **True**, HTTPS will be used as the protocol, otherwise HTTP is used.

Notes

- When set to **True**, clicking **Save** updates the URL protocol to HTTPS for only the active service. Clicking **Save to all** updates the URL protocol for all of the services that are affected by this change.

  The protocol used for the parameters listed below will be updated when the value of Enable_HTTPS changes:

  ```
  SEARCH_AUTHORITY
  VIEWER_AUTHORITY
  PUBLICATION_AUTHORITY
  MARKUP_AUTHORITY
  AUTHORITY
  MARKUP_SERVICE_ORIGIN
  MKONDO_BLOB_BASE_URL
  PUBLICATION_SERVICE_ORIGIN
  ```

- If the parameter HTTP_PROTOCOL is set to HTTPS, and values exist for any of its dependent parameter fields, those values will be cleared in the selected service env file if you set **Enable HTTPS** to **False**. In this case, upon clicking **Save**, a warning message appears to **Confirm** that you want to clear the fields for the following:

  ```
  Keystore Path
  Keystore Password
  Keystore Alias
  Keystore Alias Password
  ```

Keystore Path

Parameter name: KEYSTORE_PATH

Default value: not set

Required: YES when the HTTPS protocol is enabled (ENABLE_HTTPS).

Type: Filepath

Description: Path to the keystore JKS file string.

Keystore Password

Parameter name: KEYSTORE_PASSWORD

Default value: not set

Required: YES when the HTTPS protocol is enabled (ENABLE_HTTPS).

Type: Encrypted string

Description: Password used when creating the keystore string.

Keystore Alias

Parameter name: KEYSTORE_ALIAS

Default value: not set

Required: YES when the HTTPS protocol is enabled (ENABLE_HTTPS).

Type: Encrypted string

Description: Alias used when creating the keystore string.

Keystore Alias Password

Parameter name: KEYSTORE_ALIAS_PASSWORD

Default value: not set

Required: YES when the HTTPS protocol is enabled (ENABLE_HTTPS).

Type: Encrypted string

Description: Password set for the Alias entry within the keystore.

6.2.9 Database settings

The database settings control the communication between the Intelligent Viewing services (Configuration, Publication, Publisher, and Markup) and your database provider (PostgreSQL, Microsoft SQL Server, or Oracle).

You can set overriding database settings values for each of the individual services through the **Service Configuration** pages **Edit** buttons.

6.2.9.1 Markup Service database parameters

Database Provider

Parameter name: DB_PROVIDER

Default value: PostgreSQL

Required: YES

Type: Choice: **PostgreSQL/MS SQL/Oracle**

Description: Database server that the Markup Service should use.

Database Host

Parameter name: DB_HOST

Default value: not set

Required: YES

Type: Hostname

Description: Host name or IP for the database server that the Markup Service should use.

Database Port

Parameter name: DB_PORT

Default value: not set

Required: YES

Type: Integer

Description: Port for the database server that the Markup Service should use.

Database Name

Parameter name: DB_NAME

Default value: not set

Required: YES

Type: String

Description: Name of the database server that the Markup Service should use.

Database Username

Parameter name: DB_USER

Default value: not set

Required: YES

Type: Encrypted string

Description: Database user that the Markup Service should use.

Database Password

Parameter name: DB_PWD

Default value: not set

Required: YES

Type: Encrypted string

Description: Database password that the Markup Service should use.

Is Database configured to use SSL?

Parameter name: DB_USE_SSL

Default value: False

Required: YES

Type: Boolean

Description: Enable/disable (true/false) SSL for PostgreSQL connections.

Note: If the parameter DEFAULT_DB_USE_SSL is set to `True`, and values exist for any of its dependent parameter fields, those values will be cleared in the selected service env file if you set **Is Database configured to use SSL?** to **False**. In this case, upon clicking **Save**, a warning message appears to **Confirm** that you want to clear the fields for the following:

- Database SSL Mode
- Database Keystore Type
- Database Keystore Path
- Database Keystore Password
- Database Keystore Alias
- Database Keystore Alias Password
- Database Truststore Path
- Database Truststore Password
- Database Truststore Alias
- Database Truststore Alias Password

Database Max Pool Size

Parameter name: DB_PMAX_POOL_SIZE

Default value: 10

Required: NO

Type: Integer

Description: Maximum pool size for the database connections.

6.2.9.2 Configuration, Publication, and Publisher Service database parameters

Database Provider

Parameter name: PITHOS_PROVIDER

Default value: PostgreSQL

Required: YES

Type: Choice: **PostgreSQL/MS SQL/Oracle**

Description: Database server that the Service should use.

Database Host

Parameter name: PITHOS_HOST

Default value: not set

Required: YES

Type: Hostname

Description: Host name or IP for the database server that the Service should use.

Database Port

Parameter name: PITHOS_PORT

Default value: not set

Required: YES

Type: Integer

Description: Port for the database server that the Service should use.

Database Name

Parameter name: PITHOS_DB

Default value: not set

Required: YES

Type: String

Description: Name of the database server that the Service should use.

Database Username

Parameter name: PITHOS_USER

Default value: not set

Required: YES

Type: Encrypted string

Description: Database username that the Service should use.

Database Password

Parameter name: PITHOS_PWD

Default value: not set

Required: YES

Type: Encrypted string

Description: Database password that the Service should use.

Is Database configured to use SSL?

Parameter name: PITHOS_USE_SSL

Default value: False

Required: YES when PITHOS_PROVIDER = postgreSQL

Type: Boolean

Description: Enable/disable (true/false) SSL for PostgreSQL connections.

Notes

- **SSL support note**

  SSL support is currently available only for the PostgreSQL database.
- If the parameter DEFAULT_DB_USE_SSL is set to `True`, and values exist for any of its dependent parameter fields, those values will be cleared in the selected service env file if you set **Is Database configured to use SSL?** to **False**. In this case, upon clicking **Save**, a warning message appears to **Confirm** that you want to clear the fields for the following:
  - Database SSL Mode
  - Database Keystore Type
  - Database Keystore Path
  - Database Keystore Password
  - Database Keystore Alias
  - Database Keystore Alias Password
  - Database Truststore Path
  - Database Truststore Password
  - Database Truststore Alias
  - Database Truststore Alias Password

Database SSL Mode

Parameter name: PITHOS_SSL_MODE

Default value: disabled

Required: YES when PITHOS_USE_SSL = true

Type: Choice: prefer/require/verify-ca/verify-full

Description: Specify the desired database SSL mode as either `prefer`, `require`, `verify-ca`, or `verify-full`.

Database Keystore Type

Parameter name: PITHOS_KEYSTORE_TYPE

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Choice: `JKS/PKCS12`

Description: The type of key used in the database keystore. Can be `JKS` or `PKCS12`.

Database Truststore Path

Parameter name: PITHOS_TRUSTSTORE_PATH

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Filepath

Description: Path to truststore file to use with the database.

Database Truststore Password

Parameter name: PITHOS_TRUSTSTORE_PASSWORD

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Encrypted string

Description: Password for the truststore file provided in **Database Truststore Path**.

Database Truststore Alias

Parameter name: PITHOS_TRUSTSTORE_ALIAS

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Encrypted string

Description: Alias assigned to the certificate or key entry within your truststore.

Database Truststore Alias Password

Parameter name: PITHOS_TRUSTSTORE_ALIAS_PASSWORD

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Encrypted string

Description: Password associated with the specified alias in your truststore.

Database Keystore Path

Parameter name: PITHOS_KEYSTORE_PATH

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Filepath

Description: Path to the keystore file to use with the database.

Database Keystore Password

Parameter name: PITHOS_KEYSTORE_PASSWORD

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Encrypted string

Description: Password for the keystore file provided in **Database Keystore Path**.

Database Keystore Alias

Parameter name: PITHOS_KEYSTORE_ALIAS

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Encrypted string

Description: Alias assigned to the certificate or key entry within your keystore.

Database Keystore Alias Password

Parameter name: PITHOS_KEYSTORE_ALIAS_PASSWORD

Default value: not set

Required: YES if PITHOS_PROVIDER=postgresql, PITHOS_USE_SSL=true, and PITHOS_SSL_MODE=verify-ca/verify-full.

Type: Encrypted string

Description: Password associated with the specified alias in your keystore.

6.2.10 RabbitMQ settings

The following RabbitMQ settings are available for the Publisher Service and Publication Service.

These values can be updated individually for these services through the **Service Configuration** pages RabbitMQ **Edit** buttons.

Validation is performed only with configurations that are not using SSL (**Is RabbitMQ configured to use SSL?** = False).

This validation occurs when clicking **Save** or **Save As** in the **Edit RabbitMQ Properties For <Publisher/Publication> Service** dialog box. The connection check is made with the populated credentials and, when successful, the save action continues. If the connection is unsuccessful, the message **RabbitMQ connection has failed. Check your credentials** displays and the dialog box remains open. The message is removed when you start typing into any of the fields, in preparation for another validation check when those changes are saved. For additional tips if your RabbitMQ connection is failing when starting services, see “Service failed to start?” on page 96

RabbitMQ Server Host

Parameter name: AJIRA_HOST

Default value: localhost

Required: NO

Type: Hostname

Description: Hostname or IP of the RabbitMQ server.

RabbitMQ Server Port

Parameter name: AJIRA_PORT

Default value: 5672

Required: NO

Type: Integer

Description: AMQP port for RabbitMQ.

RabbitMQ Server Username

Parameter name: AJIRA_USER

Default value: guest

Required: NO

Type: Encrypted string

Description: RabbitMQ Username.

RabbitMQ Server Password

Parameter name: AJIRA_PWD

Default value: not set

Required: NO

Type: Encrypted string

Description: RabbitMQ Password.

Is RabbitMQ configured to use SSL?

Parameter name: AJIRA_USE_SSL

Default value: False

Required: NO

Type: Boolean

Description: Set to **True** if the RabbitMSQ server is configured to use SSL (TLS), **False** otherwise.

Note: If the parameter RMQ_USE_SSL is set to `True`, and values exist for any of its dependent parameter fields, those values will be cleared in the selected service env file if you set **Is RabbitMQ configured to use SSL?** to **False**. In this case, upon clicking **Save**, a warning message appears to **Confirm** that you want to clear the fields for the following:

- RabbitMQ Keystore Type
- RabbitMQ Keystore Path
- RabbitMQ Keystore Password
- RabbitMQ Truststore Path
- RabbitMQ Truststore Password

RabbitMQ Keystore Type

Parameter name: AJIRA_KEYSTORE_TYPE

Default value: not set

Required: YES if using SSL

Type: Choice: `JKS/PKCS12`

Description: Type of key used in the RabbitMQ keystore.

RabbitMQ Keystore Path

Parameter name: AJIRA_KEYSTORE_PATH

Default value: not set

Required: YES if using SSL

Type: Filepath

Description: Path to keystore file to use with RabbitMQ. Only used if `RMQ_USE_SSL = true`. The keystore file must be accessible to the user that the services will run under.

RabbitMQ Keystore Password

Parameter name: AJIRA_KEYSTORE_PASSWORD

Default value: not set

Required: YES if using SSL

Type: Encrypted string

Description: Password for the keystore file provided in RMQ_KEYSTORE_PATH.

RabbitMQ Truststore Path

Parameter name: AJIRA_TRUSTSTORE_PATH

Default value: not set

Required: YES if using SSL

Type: Filepath

Description: Path to truststore file to use with RabbitMQ. Only used if `RMQ_USE_SSL = true`. The truststore file must be accessible to the user that the services will run under.

RabbitMQ Truststore Password

Parameter name: AJIRA_TRUSTSTORE_PASSWORD

Default value: not set

Required: YES if using SSL

Type: Encrypted string

Description: Password for the truststore file provided in RMQ_TRUSTSTORE_PATH.

6.3 Start and stop services

You can control service startup with the Configuration Tool rather than through the Windows Services panel or Linux command prompt.

For changes in any of the service configuration files to take effect, the affected service (and any dependent services) must be restarted. You can start, stop, and restart each service from the associated service page in **Service Configuration**.

The status of the service displays on the right side of the page with options to either **Stop Service** or **Restart Service** while the service status is **Running**, or **Start Service** while the service status is **Stopped**.

If you edit any of the values in the currently selected service page, the service startup buttons remain disabled until you have saved your changes. When changes have been saved and you start a service, a message displays indicating whether or not the service configuration and startup was successful.

You can wait to restart each service after all changes have been made. Keep in mind the following service startup notes.

Service startup notes

- The order that services are started/stopped does not matter, but Windows and Linux might enforce dependencies when performing start and stop.

- If the Configuration Service is manually stopped, the Asset, Publication, and Publisher Services will also be stopped.
- You must individually start each service from the installed service pages. If you have used the **Save to all** button to save changes to all corresponding services, you will be prompted to **Start/Restart** all updated services. If you choose not to start/restart when prompted, you must manually restart each updated service later.
- Ensure that all updated services have been restarted and that all installed services are running before you exit the Configuration Tool.
- At any given point, the status of the services displayed within the Config Tool application will be the same as those displayed within the Windows Services panel or Linux command prompt.

Service failed to start?

If starting or restarting a service results in a "Failure to restart service" message, check the following troubleshooting tips.

- Services can fail to start if incorrect or unavailable settings exist in the configuration. In particular, incorrect database credentials, unavailable ports, or incorrect service origin URLs will prevent services from starting.
- All services require a valid Intelligent Viewing license.
- RabbitMQ, OTDS licensing, and your database (PostgreSQL, Microsoft SQL Server, or Oracle) must be started to run the services. If these applications are not started, the Intelligent Viewing services will not start.
- When RabbitMQ is configured without SSL, the Configuration Tool does a validation check prior to starting the Publication and Publishing Services. It fetches and validates the latest saved RabbitMQ credentials from these services `env.config` files. If the validation fails, the dependent service or actual service will not be started and a "RabbitMQ connection has failed. Check your credentials." message displays. It is recommended that you save the valid RabbitMQ credentials through the individual services forms before starting or restarting a service.
- If services are failing to start, the service logs should be reviewed for "Network Error" errors. These can occur if the service is configured with an http URI for another service, but that other service is running on an https service, or the reverse. They can also occur if a service is running as https, but the connecting service does not trust the certificate in use.
- For any service startup issues, refer to the "Service logs dashboard" on page 97. You can download and review the log files for the specific service that is failing to help determine the cause.

6.4 Service logs dashboard

The Configuration Tool provides a dashboard in which you can view and download the service log files. On the **Logs Dashboard** page, the available services are listed on the left panel and the right panel lists all of the logs of the selected service. The logs are sorted by their last modified date and time with the latest log positioned at the top of the list.

Within this panel, you can do the following:

Search logs

Point to and click on a log file to view the log data in the lower panel of the log page. To filter the displayed log entries by search keywords, type a term in the **Search** text box and click the **Search** button 🔍 (or press **ENTER**) . Only the lines containing the specified search term (if found) display in the log entries panel. Click the “x” in the search box to clear the search filter and display all log entries.

Note: Searches are case-insensitive.

Refresh logs

Click the **Refresh** button ⟳ to get the latest log files for the selected service.

Download logs

To download an individual service log file to your local file system default download directory, click the service in the **Available Services** panel and then click the corresponding **Download** button ⤓ located on the same row as the desired log filename.

To download ALL of the logs for the selected service as a single .ZIP file to your local file system, click the **Download** button ⤓ on the **Logs Dashboard** panel.

View logs

To view the latest log file content, click the specific service log filename and the content will render in a panel at the bottom of the screen within the service logs dashboard. The active log file row is highlighted to indicate for which file the content is being rendered. The log view will display up to the latest 300 entries of the file content. The bottom panel is resizable and can be closed by clicking the **Close** button.

6.5 Health dashboard

The **Health Dashboard** page provides three status tabs to quickly check the condition of the services, RabbitMQ, and databases. These checks are available to help you troubleshoot and to aid in the success of your deployment.

6.5.1 Service check

The **Service Check** dashboard lets you view and monitor the status of each of the installed Intelligent Viewing services, post configuration. Displayed on individual cards under **Transformation Services** and **Viewing Services**, you can view the hostname of the machine where the service is installed, the current status (`Running` or `Stopped`), the service version number, and (if running) the time the service was last started.

Click the dashboard's **Refresh** button to update the status of all services. If you change any parameters on other configuration tabs (such as **Service Configuration**, which stops or starts the service), refreshing the **Service Check** dashboard will fetch and display the latest information. You can refresh each service card individually by clicking the refresh button 🔄 on each card.

6.5.2 RabbitMQ check

The **RabbitMQ Check** dashboard provides visibility into the readiness of the RabbitMQ connection. This tab does not display when either SSL is enabled, or when the Publisher and Publication services are not installed.

This dashboard displays the RabbitMQ connection host and port for ease of troubleshooting. A connection status message and timestamp of the last connection check are also provided. You can click the **Refresh** button at any time to update the status information of the connection.

The RabbitMQ connection check pulls details from both the Publisher and Publication services, comparing values to use for the RabbitMQ settings AJIRA_HOST, AJIRA_PORT, AJIRA_USER, AJIRA_PWD, and AJIRA_USE_SSL. For parameter details, see "RabbitMQ settings" on page 93.

In determining the value to display for AJIRA_HOST, the RabbitMQ check uses the following logic to normalize and resolve differences:

- If both services use `localhost`, `localhost` displays.
- If one service uses `localhost` and the other points to the IP of the same machine, the IP displays.
- If one service uses `localhost` and the other points to the hostname of the same machine, the hostname displays.
- If only one service is installed, the original host value displays.
- If only one service is installed and RabbitMQ host is empty, display `Unavailable`.

If any of these parameters differ after normalization and a mismatch message displays, check that the Publication and Publisher services are using the same settings.

6.5.3 Database check

The **Database Connection Check** dashboard provides visibility into the connectivity status of all databases configured for the Intelligent Viewing services.

Notes

- The **Database check** tab displays only when the following are true:
  - All of the following services are installed: Markup, Configuration, Publisher, and Publication
  - SSL is not enabled.
- If any of the above services have SSL enabled, the health check for those services is skipped. No indication displays for the skipped services.
- For non-SSL Oracle connections, only thin mode is currently supported.

This dashboard displays a dynamic table where each row represents a unique database configuration and provides details for each database host, provider, name, port, and associated services that are using the specific database configuration.

- A green check icon in the **Status** column indicates that a successful connection exists between the dashboard and the database while a red X icon indicates a failed connection.
- The **Last Checked** column displays the time when the last connection check occurred.
- You can click the refresh button 🔄 on a database row to trigger an individual connection check. To refresh connection checks for all databases at the same time, click the **Refresh** button located at the top of the dashboard.

Chapter 7

Configure load balancer/reverse proxy

A standard production Intelligent Viewing configuration is to install the IV services configured with http communication behind a firewall, and then to front-end that with a load balancer or reverse proxy that is accessible to external end-users. Although the remainder of this section will reference “load balancer”, the same configuration applies to reverse proxies. The load balancer is typically configured to accept https connections, and then proxies requests to the IV services behind the firewall. This section describes the forwarding rules that must be established in the load balancer, as well as the extra environment configuration that is required for some of the IV services.

7.1 Routing configuration

The load balancer routing rules should match and forward the following URI paths to the back-end system that is hosing the IV services.

URI Path	Proxy Target
`/artifacts/`	`http://internal.iv.net:<AssetServicePort>/artifacts`
`/markup/api/`	`http://internal.iv.net:<MarkupServicePort>/markup/api/`
`/publication/api/`	`http://internal.iv.net:<PublicationServicePort>/publication/api/`
`/search/api/`	`http://internal.iv.net:<SearchServicePort>/search/api/`
`/viewer/api/`	`http://internal.iv.net:<ViewerServicePort>/viewer/api/`

Following is a sample `nginx.conf` file implementing the above routing rules. The ports in the `proxy_pass` arguments are the default configured ports for the various IV services. In this example, the host, `external.example.com`, is externally accessible and is listening for https connections, where as `internal.iv.net` represents the system that is hosting the IV services behind a firewall.

```
server {
    listen 443 ssl;
    server_name external.example.com;
    ssl_certificate cert.pem;
    ssl_certificate_key key.pem;
    location /artifacts/ {
        proxy_pass http://internal.iv.net:3350/artifacts/;
    }
    location /markup/api/ {
        proxy_pass http://internal.iv.net:3352/markup/api/;
    }
    location /publication/api/ {
        proxy_pass http://internal.iv.net:3356/publication/api/;
    }
    location /search/api/ {
        proxy_pass http://internal.iv.net:3357/search/api/;
    }
    location /viewer/api/ {
        proxy_pass http://internal.iv.net:3358/viewer/api/;
    }
        location / {
       proxy_pass http://internal.iv.net:3353/;
    }
}
```

7.2 IV services environmental variable configuration

The following environment variables must be configured to allow for proper operation of the IV services through a load balancer.

These environment files can be configured at any time after installation using the Configuration Tool. See “Using the Configuration Tool“ on page 69.

Publication Service

This environment config is located at `<INSTALLDIR>/PublicationService/config/env.conf`

Update the AUTHORITY environment variable to reference the load balancer URL as `https://external.example.com`, for example. Similarly, update the MKONDO_BLOB_BASE_URL variable to reference the load balancer URL as `https://external.example.com/artifacts`.

Publisher Service

This environment config is located at `<INSTALLDIR>/PublisherService/config/env.conf`

Update the MKONDO_BLOB_BASE_URL variable to reference the load balancer URL as `https://external.example.com/artifacts`, for example.

Search Service

This environment config is located at `<INSTALLDIR>/SearchService/.env`

The following environmental variables must be added:

ARTIFACT_BASE_URL

Set this variable to the URL to locally access the asset service artifacts. For example, if the Asset Service is running on the same machine as the Search Service with the default port configured, the following line would be added:

```
ARTIFACT_BASE_URL=http://localhost:3350/artifacts
```

ARTIFACT_BASE_URL_PUBLIC

Set this variable to the Asset Service artifacts URL that is externally accessible. For example:

```
ARTIFACT_BASE_URL_PUBLIC=https://external.example.com/artifacts
```

Update the SEARCH_AUTHORITY environmental variable to reference the load balancer URL as `https://external.example.com`, for example.

Viewer Service

This environment config is located at `<INSTALLDIR>/ViewerService/.env`

Update the VIEWER_AUTHORITY environmental variable to reference the load balancer URL as `https://external.example.com`, for example.

Markup Service

This environment config is located at `<INSTALLDIR>/MarkupService/.env`

Update the MARKUP_AUTHORITY environmental variable to reference the load balancer URL as `https://external.example.com`, for example.

7.3 Firewall port accessibility

If the IV services are installed on one or more machines behind a firewall, certain IV services must be accessible to the load balancer. The following lists these services along with their default ports.

Service	Default port
Asset Service	3350
Markup Service	3352
Publication Service	3356
Search Service	3357
Viewer Service	3358

The firewall rules should be updated to allow access to the configured ports for these IV services from the load balancer machine.

Chapter 8

Error logging

8.1 Types of service logging

Logging is handled differently depending on the type of service involved. The two types of services used for Intelligent Viewing logging are Java and Node.js.

Java	Node.js
Asset Service	Markup Service
Config Service	Search Service
Publication Service	Viewer Service
Publisher Service	
Config file is `env.conf`, found in the service’s `\config\` folder.	Config file is `.env` found in the root directory of the service.
Defaults are defined in `\config\wrapper.conf` (do not modify).	Services contain a `README.md` file with options information.

If you have installed the Configuration Tool, you can quickly view and download service log files through the tool. See “Service logs dashboard” on page 97.

Logs will be generated in the `/var/log/opentext/<service>` directory: After installation, `LOG_FILE_PATH=/var/log/opentext/<service>` is added to the environment configuration files (`env.config` and `.env`).

Important

- If you set a custom LOG_FILE_PATH value, you must set the value to an existing folder that can be written to by the user the service is running as. If a value is set to a path that doesn't exist, the service will not start.
- Applicable for viewing services running on Node.js: If you want to set up a custom log file location and update the value of LOG_FILE_PATH in the environment configuration file, you must also update the path in `service.xml` and then restart the respective service. Failure to do so will result in the `std.out`, `std.err`, and `wrapper.log` files not being written into the custom LOG_FILE_PATH location.

8.2 Java service logging

- The external Java logging configuration file, `logback.xml`, uses environmental variables that can be set in the `env.conf` file. For example:

  `logback.xml` setting = `<logger name=”com.opentext.service” level=”${LOG_LEVEL_SERVICE}” />`

  `env.conf` setting = `set.LOG_LEVEL_SERVICE=DEBUG`

  If using the log level set in `env.conf`, you must restart the service to apply.

  The log level can be adjusted while a service is running by modifying `logback.xml` and the new level will be picked up in approximately 30 seconds.

- If modifying `logback.xml`, best practice is to comment out and add a new line, making it simple to restore. For example:

```
<!--<logger name="com.opentext.service" level="${LOG_LEVEL_SERVICE}" />-->
<logger name="com.opentext.service" level="DEBUG" />
```

- Java logging supports setting different log levels based on the Java namespace, providing a granular level of control over the amount of logging that you are enabling for each.

- Java log files are found in the `C:\ProgramData\OpenText\logs` folder for each service.

8.2.1 Useful Java service loggers

Not all loggers are available for each service. Check the `logback.xml` or `wrapper.conf` files to verify.

Logger	When used
LOG_LEVEL_SERVICE	Debugging REST API activity.
LOG_LEVEL_ACCESS	Seeing 404 errors for publications that you think should exist.
LOG_LEVEL_EXPIRY	Debugging why temporary publications are not getting deleted.
LOG_LEVEL_SPI_PITHOS	Debugging issues with the database connection.
LOG_LEVEL_SPI_SOTERIA	Debugging OTDS authentication/ authorization issues.
LOG_LEVEL_ARTIFACTS	Debugging artifact storage and retrieval.
LOG_LEVEL_MKONDO	Debugging artifact storage and retrieval.
LOG_LEVEL_SEQUENCING	Debugging source file retrieval and publishing jobs.

8.3 Node.js service logging basics

- `Node.js` logging is configured by setting the LOG_LEVEL environment variable in the `.env` file. Valid values are: 10 = TRACE, 20 = DEBUG, 30 = INFO, 40 = WARN, and 50 = ERROR

  Any changes to logging require a restart.
- Granular logging is not available as it is with the Java services.
- If no `service.out.log` exists, the `service.wrapper.log` will likely have an error.
- `Node.js` log files are found in the `C:\ProgramData\OpenText\logs` folder for each node service.

8.4 Log reading tips

- When reviewing log files, work from the bottom up. The most recent errors are the most relevant.
- Use the available tools to make reading logs easier.

  For example, tools that can be used to read logs include Notepad++, TextPad, BareTail, mTail.
- All services make use of log levels. Use search/filter/highlighting for common terms such as *exception*, *critical*, *error*, and *warn* in your log reading tool of choice.
- Keep in mind that most services will have errors and warnings that are expected during startup that might distract you from the issue that you are attempting to troubleshoot.

Chapter 9

Troubleshooting

9.1 General tips

- Verify the versions of the prerequisites used with Intelligent Viewing are the supported versions specified in the release notes.
- Be sure to run the installer from an administrator account (“Run as Administrator”).
- API, Viewing, and Transformation Service documentation and tutorials are available from the OpenText Developer network **Products > Viewing & Transformation Services** page:

  https://developer.opentext.com/services/products/viewing-transformation-services

  This documentation applies to cloud services (Core Viewing Service, Core Transformation Service) as well as Intelligent Viewing.
- The Viewer Service or Publisher Service are the most likely points of failure due to their complexity when compared with the rest of the services.
- Be aware that the viewer user interface is very modular and highly customizable in regard to the available tools and panels, and will vary widely between integrations.
- The only client log available for troubleshooting is through the browser’s developer console (**F12** > **Console** tab). Additionally, the following tabs of the dev tools can be used for troubleshooting purposes:

  Network – used for inspecting requests and responses.

  Console – used to see if the viewer or integration is throwing any warnings or errors.

  Inspector – used for issues with the viewer layout, CSS, and for looking at the page source.

  Performance – used for debugging performance issues.
- If you are configuring SSL with IV, verify the following:
  - The keystore is created and configured correctly.
  - The SSL certificate trust has been established between IF and the integration.

9.2 Service check

When the installation has completed, check the health of the services using the URLs listed in the following table (replacing <FQDN> with the appropriate fully qualified domain name). If they are up and running, the installation is successful.

Service	URL to check the health of the service
Config	`http://<FQDN>:3351/config/api/v1/health`
Markup	`http://<FQDN>:3352/markup/api/v1/health`
Publication	`http://<FQDN>:3356/publication/api/v1/health`
Search	`http://<FQDN>:3357/search/api/v1/health`
Viewer	`http://<FQDN>:3358/viewer/api/v1/health`
Publisher	`http://<FQDN>:9092/publisher/api/v1/health/live`

9.3 Publishing tips

- Chrome instances that are launched by the Publisher Service to process HTML files for viewing run under tight security restrictions. During the IV installation, a URLBlocklist is installed for the Chrome web browser that prevents Chrome from accessing http, https, and file URLs. This is a security measure to prevent any malicious files that are processed through IV from accessing locations they should not have access to. Due to this Chrome restriction, users cannot use Chrome from that server in any useful fashion (such as using the server to download an updated version of a required component) because only the Publisher Service has access to use Chrome.
- If the source file that the Publisher Service is retrieving for conversion is at an http URL, by default no authorization header will be sent with the request from the Publisher Service. If your file server requires an authorization header, the request from the Publisher Service will fail and the logs will report that the request was made without an Authorization header. To address this, add the URL of the server that is serving the files to the TRUSTED_SOURCE_ORIGINS environment variables in the Publisher and Publication Services `env.conf` files. See "Updating the IV service configuration" on page 67.

9.4 Installer error messages

This section provides possible causes and solutions for error messages that you might encounter during your installation.

Unable to connect to RabbitMQ

Causes

- Intelligent Viewing and RabbitMQ versions are incompatible.
- ErLang OTP with respect to RabbitMQ version is incompatible.
- RabbitMQ plug-ins are not enabled.

Solutions

- Verify that the RabbitMQ version matches the requirement specified in the release notes.
- Verify that the Erlang OTP version is compatible with the RabbitMQ version.
- The default `user:password` for RabbitMQ is `guest:guest`. Because this user account is limited to localhost, you are required to create a new user for remote host connections. To create a different user account, use the command:

  ```
  rabbitmqctl.bat add_user <username> <password>
  ```

- Verify that RabbitMQ is accessible using the URL: localhost:15672.
- If the URL is not accessible, enable the `rabbitmq_management` plug-in by running the following command from the location: `ProgramFiles/RabbitMQ/sbin`.

  ```
  rabbitmq-plugins.bat enable rabbitmq_management
  ```

  Restarting the RabbitMQ service after the above changes will enable the URL. If it remains inaccessible, clear the browser cache or try again with your browser’s Incognito mode enabled.

Unable to retrieve OTDS version info

Causes

- Intelligent Viewing and OTDS versions are incompatible.
- The installation is interrupted.
- Improper OTDS credentials are provided in the `IntelligentViewing_MSI.properties` file.

Solutions

- Verify that the OTDS and Intelligent Viewing versions are compatible.

- Verify that OTDS is running before starting the Intelligent Viewing installation.
- Verify the accuracy of the OTDS parameters OTDS_ORIGIN, OTDS_ADMIN_USER, and OTDS_PW provided in the `IntelligentViewing_MSI.properties` file.
- Although not recommended, in certain scenarios, doing a clean uninstall and reinstall can help.

There was an unknown error when running OTDSConfig.exe

Causes

- Intelligent Viewing and OTDS versions are incompatible.
- Improper OTDS credentials are provided in the `IntelligentViewing_MSI.properties` file.
- Improper JDBC connection string details provided during the OTDS installation.

Solutions

- Verify that the OTDS and Intelligent Viewing versions are compatible.
- Verify the accuracy of the OTDS credentials OTDS_ORIGIN, OTDS_ADMIN_USER, and OTDS_PW. The OTDS_ORIGIN format should be `http://<hostname>:<port>/otdsws`
- Verify that the correct JDBC connection string was provided during the OTDS installation.

IV services are not up and running

Causes

- This message is seen after the installation if the `IntelligentViewing_MSI.properties` file is not configured properly.

Solutions

- Verify that the `IntelligentViewing_MSI.properties` file values are configured correctly and that the URLs and hosts provided are accurate.
- Verify that DEFAULT_HOST is configured with FQDN and not localhost.
- Verify that all the services are running under an account with admin privileges.

Part 3

Security Implementation

Chapter 10

Introduction

This document provides information for implementing security hardening of Intelligent Viewing and presents techniques you can use to make deployments more resilient to attack. Its purpose is to help you, as an administrator, determine which configuration options will meet the needs of business users as well as meet the security requirements and risk tolerance of your organization.

Intelligent Viewing works in conjunction with infrastructure service software and hardware such as server operating systems, load balancers, IDSs, databases, and web servers. This document focuses solely on the configuration of the Intelligent Viewing application. The configuration of supporting software, services, and servers should be reviewed to ensure that Intelligent Viewing security is supported by secure supporting systems. We recommended that each infrastructure component be hardened as per the best practices of that vendor.

Chapter 11

Intelligent Viewing risk assessment

11.1 Operating system

Intelligent Viewing is supported on Windows Server 2019 and 2022. Risks associated with operating systems include security vulnerabilities, malware threats, unauthorized access, data breaches, and denial–of–service attacks.

11.2 Credential and configuration management

Intelligent Viewing stores credentials and other configuration information in files that are written to disk. If an unauthorized user gains access to the file system, they can potentially retrieve sensitive credentials and configuration data. Sensitive information can be inadvertently exposed if files are shared or backed up without proper security measures.

11.3 Databases

Intelligent Viewing stores configuration and publication details in the database. Configuration data includes the metadata needed for processing and validating publication details such as application, policy and features. The publication data includes transactional data created for identifying and tracking the publications.

Database can be exploited through various techniques and vulnerabilities that attackers can target to gain unauthorized access, manipulate data, or disrupt services.

11.4 RabbitMQ

Intelligent Viewing uses RabbitMQ for queuing publication requests. The requests are queued by the Publication service and then consumed by the Publisher service.

RabbitMQ risks include unauthorized access, data leakage, message tampering, unpatched software, unauthorized code execution, and lack of encrypted communication.

11.5 Java

Some of the Intelligent Viewing services run a Java virtual machine. Java risks include remote code execution, code injection, de-serialization vulnerabilities, insecure libraries, XSS attacks, SQL injection, inadequate authentication, and insufficient input validation. These risks highlight the need for secure coding, regular updates, and comprehensive monitoring.

11.6 Node.js

NPM is a package manager for Node.js. Some of the Intelligent Viewing services are written in JavaScript and run on Node.js, a cross-platform back-end JavaScript runtime environment.

11.7 REST end points

Intelligent Viewing exposes various REST end points from different services to be consumed by different products. REST endpoints risk exposure to injection attacks, insecure authentication/authorization, broken access controls, XSS, SQL injection, DoS, information disclosure, mass assignment, lack of validation, and insufficient logging.

11.8 OTDS

Open Text Directory Services (OTDS) is a repository of user and group identity information and uses a collection of services to manage this information for OpenText applications. OTDS contains components for identity synchronization and single sign-on for all OpenText applications.

Intelligent Viewing uses OTDS for Authentication and Authorization.

Chapter 12

Best practices and security hardening

12.1 Operating systems

Operating systems can be vulnerable to malware threats, unauthorized access, data breaches, and denial–of–service attacks if security measures are not properly implemented. Refer to your operating system documentation for additional information on security hardening practices.

Mitigation strategies

To mitigate risk of operating system exploitation, follow these best practices:

Encryption methods:
Encrypt a file system at rest. The following options are available for Windows.

See also "Security settings" on page 28.

- **BitLocker**: BitLocker is a built-in encryption feature with Windows. It can encrypt the entire system drive or specific data drives. For details, see https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/.
- **EFS (Encrypting File System)**: EFS allows you to encrypt individual files and folders. For details, see https://learn.microsoft.com/en-us/windows/win32/fileio/file-encryption?redirectedfrom=MSDN.

Patch management:
Keep the operating system up to date with the latest security patches and updates.

Authentication methods:
Implement strong password policies and consider multi-factor authentication (MFA) for enhanced security. Use complex, unique passwords for administrator accounts.

Permissions management:
Assign roles and permissions to users and groups based on the principle of least privilege (POLP). Ensure that users have only the minimum level of access necessary to perform their tasks.

Audit policies:
Set up auditing policies to track and log security events. Regularly review and analyze logs for signs of unauthorized access or suspicious activities.

Firewall protection:
Use Windows Firewall or a third-party firewall solution to control incoming and outgoing network traffic. Restrict access to necessary ports and services.

Backup policies:
Implement regular backup and disaster recovery plans to ensure data integrity and availability. Test backups regularly to verify they can be restored successfully.

Antivirus policies:
Install and regularly update antivirus and anti-malware software. Configure real-time scanning and scheduled scans to detect and remove threats.

Essential services:
Disable any unnecessary Windows features and services that are not essential for the server's function. This reduces the attack surface.

Remote Desktop Service:
If using a Remote Desktop Service, secure it by configuring Network Level Authentication (NLA) that enables strong authentication and limits access to trusted users.

12.2 Credential and configuration management

Credential and configuration files can be vulnerable to security exploitation if security measures are not properly implemented.

Mitigation strategies

To mitigate risk of credential and configuration file exploitation, follow these best practices:

Implement POLP:
Implement a principle of least privilege (POLP), ensuring that only necessary personnel have access to the Intelligent Viewing configuration files.

Monitor activities:
Monitor file access and usage for suspicious activities.

Secure channels:
Avoid sharing credentials and configuration files through insecure channels.

Encryption management:
Use encryption for backups and transfers to prevent data exposure.

12.3 Databases

Databases can be vulnerable to various forms of exploitation if security measures are not properly implemented. Common ways that databases can be exploited include the following:

Unauthorized access:
 Weak authentication mechanisms, default credentials, or misconfigured access controls can allow attackers to gain unauthorized to the database.

Vulnerabilities in database software:
Unpatched or outdated database software can have known vulnerabilities that attackers exploit to gain access or control over the database,

Lack of encryption:
Without proper encryption, attackers can intercept data in transit or access sensitive data at rest.

Mitigation strategies

To mitigate the risk of database exploitation, follow these best practices:

Authentication methods:
Strengthen authentication methods, enforce strong password policies, remove default credentials, regularly review and update access controls, and conduct security audits to prevent unauthorized database access.

Patch management:
Establish a proactive patch management process, regularly update database software, and closely monitor security advisories to mitigate vulnerabilities and reduce the risk of exploitation.

Encryption enforcement:
Enforce end-to-end encryption for data in transit using secure protocols (for example, TLS/SSL), implement encryption for data at rest, utilize strong encryption algorithms, and manage encryption keys securely to safeguard sensitive information from unauthorized access.

Configure SSL:
Intelligent Viewing supports SSL communication for the PostgreSQL database.

See “Security settings” on page 28.

SSL support note

SSL support is only available for the PostgreSQL database.

Additional PostgreSQL SSL configuration information can be found at https://www.postgresql.org/docs/current/ssl-tcp.html.

12.4 RabbitMQ

RabbitMQ can be vulnerable to various forms of exploitation if security measures are not properly implemented. Common ways that RabbitMQ can be exploited include the following:

Unauthorized access:
Weak authentication or misconfigured access controls can allow unauthorized users to gain access to RabbitMQ.

Data leakage:
Sensitive data can be exposed if message queues are not properly secured.

Message tampering:
Attackers can intercept and modify messages in transit, potentially leading to data integrity issues.

Injection attacks:
Malicious users can inject unauthorized code into messages, compromising the integrity of the system.

Unpatched software:
Running outdated RabbitMQ versions can expose the system to known vulnerabilities.

Lack of encrypted communication:
Unencrypted communication between RabbitMQ nodes or clients can expose sensitive data.

Insecure authentication and authorization:
Weak or improperly implemented authentication and authorization mechanisms can allow unauthorized access to sensitive functions or data.

Mitigation strategies

To mitigate the risk of RabbitMQ exploitation, follow these best practices:

Authentication methods:
Implement strong authentication mechanisms, enforce access controls, and follow the principle of least privilege. Regularly review user permissions.

Audit settings:
Regularly review and audit RabbitMQ configuration settings to ensure they align with security best practices.

Patch management:
Keep RabbitMQ up to date with the latest security patches and updates.

SSL encryption:
Enable SSL/TLS encryption for communication between RabbitMQ components and clients.

Additional RabbitMQ configuration information can be found at https://www.rabbitmq.com/ssl.html.

12.5 Java/Node.js

Java and Node.js are widely used programming languages that can be vulnerable to exploitation if security weaknesses or vulnerabilities are present in the code, runtime environment, or application design. Ways that Java can be exploited include the following:

Remote Code Execution (RCE):
Attackers can exploit vulnerabilities in Java applications to execute arbitrary code remotely, potentially gaining control over the target system.

Code injection:
Malicious code can be injected into Java applications through user inputs, leading to unintended execution of unauthorized actions.

Deserialization vulnerabilities:
Attackers can manipulate serialized Java objects to execute malicious code during deserialization, potentially leading to RCE.

Insecure libraries:
Use of outdated or vulnerable third-party libraries in Java applications can introduce security flaws that attackers can exploit.

Cross-Site Scripting (XSS):
Java-based web applications can be vulnerable to XSS attacks, where attackers inject malicious scripts that are executed by users' browsers.

Insecure deserialization:
Poorly validated deserialization of data can lead to attackers crafting malicious input to exploit vulnerabilities.

SQL injection:
Java applications using databases may be vulnerable to SQL injection attacks if user inputs are not properly sanitized.

Insecure authentication and authorization:
Weak or improperly implemented authentication and authorization mechanisms can allow unauthorized access to sensitive functions or data.

Insufficient input validation:
Failure to properly validate and sanitize user inputs can lead to a variety of security vulnerabilities.

Inadequate logging and monitoring:
Lack of comprehensive logging and monitoring can hinder the detection of unauthorized activities and breaches.

Mitigation strategies

To mitigate the risk of Java exploitation, follow these best practices:

Regular updates:
Keep the Java runtime environment and libraries up to date to gain security patch benefits.

Third–party libraries:
Only use reputable and up-to-date third-party libraries, updating them regularly.

12.6 REST endpoints

REST endpoints can be vulnerable to various forms of exploitation if security measures are not properly implemented. Common ways REST endpoints can be exploited include the following:

Insecure authentication and authorization:
Weak or misconfigured authentication mechanisms can allow unauthorized access to sensitive resources, while inadequate authorization controls can lead to improper access rights.

Data interception:
Without SSL/TLS, data transmitted between the client and the server is sent in plain text. This makes it vulnerable to interception by attackers, potentially exposing sensitive information such as login credentials, personal data, and financial details.

Mitigation strategies

To mitigate the risk of REST end point exploitation, follow these best practices:

Control access:
Configure access controls accurately to ensure authorized users have appropriate permissions

Enable SSL/TLS
Enable SSL/TLS to access rest end points over secure channels.

Configure SSL:
To configure SSL on the Intelligent Viewing services, see:

"Security settings" on page 28.

Chapter 13

Software updates

13.1 OpenText software

It is essential that all of the latest OpenText application updates and patches are applied to ensure that an Intelligent Viewing install has all the latest security fixes. We recommend that you use OpenText System Center to apply all appropriate upgrades/patches. All upgrades and patches should also be applied for any other OpenText software deployed as part of the Intelligent Viewing system.

13.2 Ecosystem software

It is also essential that the latest updates and patches are applied for all components of the Intelligent Viewing ecosystem. This includes the platform that directly interacts with Intelligent Viewing – operating system, database, and web server – as well as other infrastructure items such as Load Balancers and Firewalls.

Chapter 14

External elements

14.1 Server infrastructure

Intelligent Viewing relies on underlying systems such as operating systems, file servers, web servers, and databases. All of these systems should be hardened according to the vendor's best practices. All third-party software should have the most recent security patch sets and updates applied. It is also advisable to use third-party products for additional protection of the network infrastructure. Web Server(s) must implement hardening best practices and protection mechanisms including adequate protocols, cluster management, failover, and filtering of incoming connections.

14.2 Client infrastructure

Client systems should be hardened according to the operating system vendor's specifications and security best practices applied. Supported browsers should be on the latest security patches.

14.2.1 Anti-virus and malware considerations

The term "virus" is used to refer to various kinds of malware including worms, Trojan horses, logic bombs, and other malicious and/or invasive pieces of code within corporate or organizational environments. Administrators must ensure that all components of the OpenText Intelligent Viewing ecosystem are protected against malicious files and scanned for viruses on a regular basis. Antivirus applications can have detrimental effects on application deployments if not configured correctly.

- An antivirus application must be able to read files from the file system. Sometimes, the process of reading the file will "lock" it and prevent other applications from reading from or writing to the file.
- When an antivirus application has detected what it believes to be an infected file, the antivirus application can (depending on configuration) "quarantine" (move and restrict access) the file or delete the file entirely.

Antivirus applications can sometimes identify non-malicious files as a virus. As a result, these files can be inadvertently deleted or quarantined. When this happens, important files can be made inaccessible that can prevent services from running correctly or from running at all.

As an administrator, you should be aware that Intelligent Viewing can be deployed in a variety of configurations - including across multiple servers, which may use different operating systems. Administrators can also enhance the default Intelligent Viewing services with add-ons and customizations. We recommend that you test changes to configuration of antivirus applications prior to deploying them in production.

14.2.2 Client antivirus protection

Ensure that any client system connecting to Intelligent Viewing has real-time virus protection, and that virus definitions are regularly updated. This is the first line of defense to counter virus threats in your corporate or organizational environments. Administrators should consult their antivirus software vendor’s website for up-to-date information, critical updates, and patches to ensure their corporate or personal virus scanners have the latest fixes to deal with any reported issues. OpenText Partner modules integrate directly with Intelligent Viewing and an organization’s existing antivirus solutions to block malware and viruses from being uploaded to Intelligent Viewing. To inquire about these solutions, please contact your Customer Support channel.

14.3 Server antivirus protection

14.3.1 Database

Databases can also have their own recommendations and guidelines regarding virus scanning of their application folders. Administrators should consult the latest published documentation by the respective database and virus scanning vendors. When an antivirus application performs a scan on a file, it places a lock on it. A file lock interrupts the normal functioning of a database. To prevent situations such as a database crash or hang, we recommend that the corresponding database files are excluded from antivirus scanning.

14.3.2 Scan timing

We recommend that antivirus scanning be performed in “scan-on-write” fashion for files and directories not specified to be omitted from antivirus scans. Most antivirus applications have three options as to when the a scan of a file is triggered:

- On-demand or scheduled inspection of files in a file system.
- Scan-on-write – Files are inspected when they are written to the file system.
- Scan-on-read – Files are inspected when they are read from the file system.

If “on-demand” or scheduled scanning takes place when the system is operating, files can be locked by the antivirus application. If the “scan-on-read” option is applied to installation directories for components of Intelligent Viewing, performance will be degraded for some operations.

14.3.3 Scan folders

We recommend that you exclude certain folders from the scan. Because log files are constantly being manipulated and overwritten by Intelligent Viewing, scanning with antivirus software, which results in files being locked, will cause issues with the services. The following directories should not be scanned by antivirus applications due to this issue:

- `\PublicationService\logs`
- `\ConfigService\logs`
- `\PublisherService\logs`
- `\AssetService\logs`
- `\MarkupService\wrapper`

Part 4

Sizing and performance tuning

Chapter 15

Sizing recommendations for peak performance

This chapter provides general sizing recommendations for your Intelligent Viewing deployment at your on-premise datacenter or virtualized cloud environment.

The sizes shown in the table below are detailed in the sample configurations (see “Sizing configurations” on page 135). These sizes are only broad-based indicators. Administrators can arrive at an appropriate sizing estimation by using these ranges while considering the information that follows.

Size	Number of concurrent users
Small	20
Medium	100
Medium Large	250
Large	500
Extra Large	1000+

15.1 Assumptions

The configuration recommendations assume the following types of usage for all sizing scenarios:

- Documents are more often viewed than published. This means that after the initial ingestion process, new documents are added to the system at a slower rate and existing documents will be viewed more often.
- Twenty percent of the user count will need concurrent publishing bandwidth. By default, each VM accommodates 9 active concurrent documents.

By default, the Publisher is configured to have 3 job queues: one for Microsoft Office documents, a second for PDF documents, and a third for CAD drawings and all other supported formats. Each queue has a maximum concurrency level of 3 jobs from each queue being published at the same time, totaling 9 documents concurrently.

Wherever the Intelligent Viewing ecosystem has more than 9 processor cores, the queue sizes can be adjusted accordingly to make use of processor cores.

The configuration for the queue sizes for various document types is set in the following publisher configuration file (restart is not required): `(PUBLISHER_HOME)/publisher.properties`

Example 15-1:

Total Processor cores: 16 (all Publisher VMs combined)

Doc type	Percentage
Office	20%
PDF	50%
DRW*	30%

*Includes image types and other supported documents not included in Office or PDF type.

The queue configuration can be set as follows:

```
queues.doc=5

queues.pdf=8

queues.drw=3
```

15.2 Publishing considerations

Publishing refers to the conversion of documents of various formats (PDF, Microsoft Office, email, CAD) to a platform/browser-neutral, portable, and scalable image format.

This conversion is internally handled by a process called Converter. The Converter is a single-threaded application that uses only one core of the CPU. Therefore, if a machine where the Publisher is installed has a 4-core CPU, 4 instances of the Converter process can publish 4 documents in parallel without any wait time.

Furthermore, if 6 documents are submitted to the Publisher running on a 4-core CPU, 4 documents will be immediately taken up for publishing and the remaining 2 documents will be queued in a messaging application (RabbitMQ) and will be published after the first batch of 4 documents is finished.

This can have a direct implication on the user experience as the users of the last 2 documents will experience latency before they are able to see the contents of their documents. This can be mitigated through one of two methods:

1. Vertical scaling: Increasing the number of CPU cores on the VM where Publisher is installed.
2. Horizontal scaling: Adding more Publisher instances to the Intelligent Viewing installation ecosystem. For more information, see *OpenText Intelligent Viewing - High Availability Install Guide (CLIVSA-IHA)*.

It is also worth noting that publishing is usually a one-time activity. After the documents are published, the resulting artifacts are saved on disk and subsequent attempts to view the same document do not require any publishing. Instead, the artifacts previously created are used. This is especially true in the case of publishing for viewing as opposed to publishing for exporting to PDF or TIFF.

When determining appropriate sizing estimations, this is an important factor to consider in order to prevent over-sizing or under-sizing.

15.3 Sizing configurations

The following sizing configurations are provided as recommended examples and can be adjusted as needed for your environment.

Table 15-1: Small: 20 concurrent users (scenario 1)

Hardware	Configuration
CPU	1 CPU with 8 cores
RAM	16 GB
Disk	120 GB
IV full installation (all services)	1
Additional Publisher-only VMs	0
Total VMs	1

Table 15-2: Small: 20 concurrent users (scenario 2)

Hardware	Configuration
CPU	1 CPU with 4 cores
RAM	16 GB
Disk	120 GB
IV full installation (all services)	1
Additional Publisher-only VMs	1
Total VMs	2

Table 15-3: Medium: 100 concurrent users (scenario 1)

Hardware	Configuration
CPU	1 CPU with 16 cores
RAM	16 GB
Disk	250 GB
IV full installation (all services)	1

Hardware	Configuration
Additional Publisher-only VMs	0
Total VMs	1

Table 15-4: Medium: 100 concurrent users (scenario 2)

Hardware	Configuration
CPU	1 CPU with 8 cores
RAM	16 GB
Disk	250 GB
IV full installation (all services)	1
Additional Publisher-only VMs	1
Total VMs	2

Table 15-5: Medium-Large: 250 concurrent users (scenario 1)

Hardware	Configuration
CPU	1 CPU with 16 cores
RAM	16 GB
Disk	250 GB
IV full installation (all services)	2
Additional Publisher-only VMs	0
Total VMs	2

Table 15-6: Medium-Large: 250 concurrent users (scenario 2)

Hardware	Configuration
CPU	1 CPU with 8 cores
RAM	16 GB
Disk	250 GB
IV full installation (all services)	2
Additional Publisher-only VMs	2
Total VMs	4

Table 15-7: Large: 500 concurrent users (scenario 1)

Hardware	Configuration
CPU	1 CPU with 16 cores

Hardware	Configuration
RAM	16 GB
Disk	500 GB
IV full installation (all services)	3
Additional Publisher-only VMs	0
Total VMs	3

Table 15-8: Large: 500 concurrent users (scenario 2)

Hardware	Configuration
CPU	1 CPU with 8 cores
RAM	16 GB
Disk	500 GB
IV full installation (all services)	3
Additional Publisher-only VMs	3
Total VMs	6

Table 15-9: Extra-Large: 1000+ concurrent users (scenario 1)

Hardware	Configuration
CPU	1 CPU with 16 cores
RAM	16 GB
Disk	1 TB
IV full installation (all services)	5
Additional Publisher-only VMs	0
Total VMs	5

Table 15-10: Extra-Large: 1000+ concurrent users (scenario 2)

Hardware	Configuration
CPU	1 CPU with 8 cores
RAM	16 GB
Disk	1 TB
IV full installation (all services)	5
Additional Publisher-only VMs	5
Total VMs	10

Chapter 16

Performance recommendations

Before discussing best practices for peak performance, it is important to understand the process by which documents are transformed and viewed. Review the services overview section, “IV service overview” on page 9, to become familiar with the product’s architecture and how Intelligent Viewing transforms and views documents, and then return to this section.

16.1 How document transformation and viewing works

The following sets of steps illustrate the sequence of interactions between the client and the Intelligent Viewing services that result in documents being transformed and viewed.

16.1.1 Creating publications

1. The client issues a POST request to the Publication Service with a body specifying the publishing profile to use and any feature settings. For information about the available publishing profiles and features, see the Intelligent Viewing developer documentation: https://developer.opentext.com/imservices/products/viewingtransformationservices
2. The Publication Service forwards the requested profile and features/settings data to the Configuration Service, which loads profile settings, adds feature settings, and validates the results against policy requirements and feature schemas.
3. If validated, the Configuration Service generates a new config object and writes it to persistent storage.
4. The config object is returned to the Publication Service.
5. The Publication Service merges the Configuration Service results with the publication JSON data and writes the result to persistent storage, generating a new Publication ID.
6. The Publication Service returns the publication JSON containing its ID, with an initial status of “Pending”. At this point, the client can poll the service to monitor the publishing status by issuing GET requests to `/publication/api/v1/publications/{id}`
7. The client then submits this publication to the viewer by calling the *`addPublication`* method. The viewer can poll the GET publication until artifacts are available to view.

16.1.2 Initiating views

After the publication is created and the response is returned to the client, the publishing process starts in the background to produce the artifacts as outlined below:

1. The Publication Service creates a directory in a configured publication artifact storage volume (sometimes referred as a blob storage) named with the publication ID.
2. The Publisher enqueues the publishing request for the next available Publisher Service to process.
3. An available Publisher checks its job queue for new requests.
4. The queue delivers the new publishing request to the Publisher.
5. The Publisher sends a notification indicating publishing has started. The Publication Service receives this notification and updates the publication's status to “Active”.
6. The Publisher issues an HTTP GET request to retrieve the requested source document(s) from the hosting service, providing its *OAuth2* service token as a bearer token in the authorization header for authentication.
7. The document hosting service authenticates the publisher's bearer token and approves the download request(s).
8. The Publisher loads the source document(s) and starts the transformation.
9. As soon as the first page is transformed, and each time new pages are generated, the publisher writes the generated artifacts to the storage volume under the publication folder.
10. For the first page's artifacts, and for all future updates during the transformation, the Publisher sends a notification describing all of the generated artifacts and the asset-service URLs for each artifact. The Publication Service receives this notification and updates the publication in the persistent storage with this data on each update.
11. When the transformation finishes (either successfully or unsuccessfully), the Publisher sends a notification indicating the process has finished. The Publication Service receives this notification and updates the status to either “Complete” or “Failed”.
12. After the publication is “Active”, the client can retrieve any artifacts that have been reported thus far by issuing an HTTP GET request to the Publication Service and retrieving the artifact URLs that have been generated.
13. The client's Authorization request header is forwarded to the document repository where the publication's source document is hosted to verify that the client should be granted access to renditions of the content
14. The document repository authenticates client credentials and evaluates access and responds with 200 OK for approval; otherwise, it responds with a 403 error.

15. The Publication Service returns the publication JSON, which includes the Asset Service download URLs for artifacts generated for the source document.
16. The Asset Service verifies the client's access to the artifact and reads it from the artifact storage.
17. The Asset Service streams the artifact bytes back to the requesting client.

16.1.3 Potential bottlenecks for the publication workflow

1. The Publisher downloads files from the source repository (see “Initiating views” on page 140, step 8). If the source repository’s API is slow, the publishing process can take too long, delaying the ability of the viewer to display the first page. Furthermore, the publication also calls the source repository to validate the permissions to access the document (see step 14). Therefore, the performance of Intelligent Viewing heavily depends on network speed and the response time of the source document access API.
2. Change of the publication status from *Pending* to *Active* depends on the number of publishing job processors available for each document type. The wait time for processing jobs will increase if the load is high and there aren't enough publishing job processors available to handle the publication jobs in the queue.
3. Artifact caches are often located on shared network storage. Any slowness in network I/O between the Publisher host and the network storage host can increase the processing time for each job, which can eventually slow down queue processing and increase the wait time for jobs in the queue.

16.2 Recommendations

From a performance perspective, all Transformation Services require fast Input/Output infrastructure (both network and filesystem), with a strong emphasis on latency and throughput to the publication artifact storage. The CPU is used mainly for processing HTTP requests and handling events to update the publication status. While converting documents, the Publisher requires many CPU cycles.

For CPU and memory requirements based on the deployment method and expected needs, see “Sizing recommendations for peak performance“ on page 133.

16.2.1 Resource impact

The following tables list each service's dependency level on each impacted resource.

Table 16-1: Transformation services

Resource	Configuration	Publication	Publisher	Asset
CPU	Moderate	High	High	Low
Memory	Low	Moderate	High	Low
Network	Low	High	High	Moderate

Resource	Configuration	Publication	Publisher	Asset
Disk	Low	Moderate (Write)	High (Write)	High (Read)

Table 16-2: Viewing services

Resource	Viewer	Markup	Highlight
CPU	Low	Low	Low
Memory	Low	Low	Low
Network	Low	Moderate	Low
Disk	Low	Low	Low

16.2.2 General considerations for optimal performance

To improve reliability of Intelligent Viewing under heavy load, consider separating the critical dependent services (the database, RabbitMQ, and OTDS) onto isolated infrastructures. When these supporting services are not isolated, they can become overloaded and unresponsive, impacting the entire Intelligent Viewing application and potentially causing it to crash.

Cloud deployments use containers to isolate services, which can improve performance and scalability. On Windows and Linux, deploying everything on a single machine can impact performance if resources become constrained, and therefore higher consideration to deploy resource-intensive services on to a separate machine is recommended (see “Intelligent Viewing Service deployment options” on page 142).

16.2.3 Intelligent Viewing Service deployment options

Depending on the hardware resources that you have available, two main deployment configurations for the IV services are recommended:

Single machine (for high capacity environments)

Install all Intelligent Viewing services on one machine.

In cases where the infrastructure is powerful enough (for example, a CPU with at least 32 cores and at least 32 GB of RAM), it is possible to deploy all of the Intelligent Viewing services, including the Publishing Agent, in the same machine, and still have a performant application.

Multiple machines (distributed Intelligent Viewing services)

Install Intelligent Viewing services on two or more separate machines: One machine dedicated to the resource-intensive Publishing Agent and all remaining services deployed in the other.

For example, provided that there are two machines available, each with 16 CPU cores and 16 GB of RAM, one may be dedicated solely to the Publishing Agent and the second one hosting the rest of the Intelligent Viewing services.

Important

Separating the Publisher Service requires additional cluster configuration. For more information, see *OpenText Intelligent Viewing - High Availability Install Guide (CLIVSA-IHA)*.

16.2.4 Adjust the default configuration

The Transformation Services use the Java runtime. By default, these services can use different initial minimum (-Xms) and maximum (-Xmx) heap size settings for the test environment to run on lower-configuration machines. In a production environment, product administrators should adjust these default -Xms and -Xmx values in the `conf/wrapper.conf` file. Typically, both -Xms and -Xmx should be set to the same value (based on available RAM) to reduce garbage collection overhead.

16.2.5 Log level settings

Different services of Intelligent Viewing use logging to provide information that can be used to track down the root cause of a problem. To reduce overhead of CPU and disk space, debug level logging is not enabled by default in production. System administers can set different log levels for the various services by editing the environment variables in `env.conf` for Windows and Linux. For Kubernetes deployments, helm properties need to be set and the container redeployed.

16.2.6 Publication artifact storage volume

The speed of storage disks directly affects how quickly publication artifacts are created and accessed. For faster deliveries and smoother access, using Solid State Drives (SSDs) is recommended. Additionally, if your Intelligent Viewing deployment is spread across multiple machines, dedicating a separate storage volume for publication artifacts can be beneficial for centralized management and easier scaling in the future.

16.2.7 Monitor and adjust doc type threads

The level of performance depends on how quickly the Publishing agent takes jobs from the queue. When publications continue to be created and the available Publishers cannot process the jobs as fast as they come in, the wait time in the queue is increased and performance suffers as a result. To decrease the job wait time in the queue, you must first understand the main doc type queues and how to monitor the number of jobs that are waiting or processing.

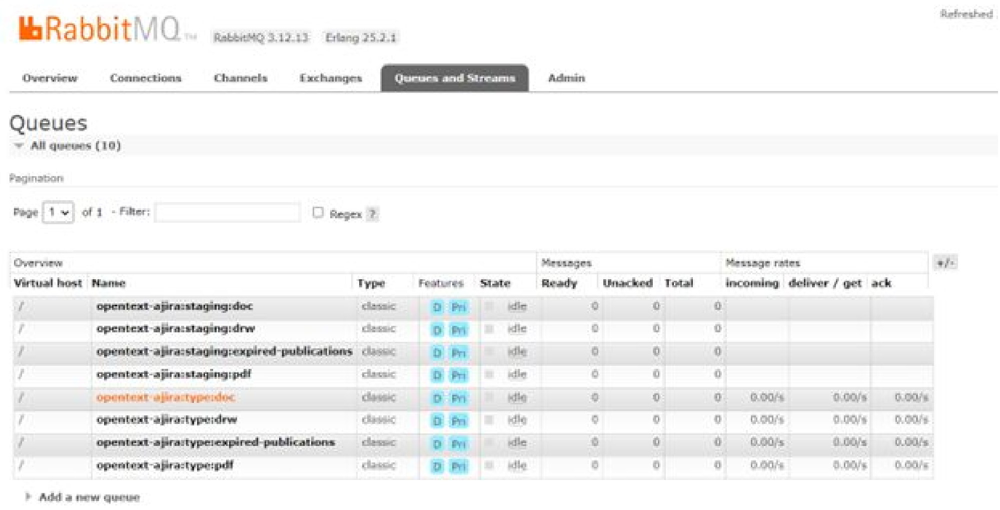
The RabbitMQ management tool, available through a user interface or command line, provides the ability to monitor the status of the jobs in the queue. You can view helpful information needed to adjust the number of Publisher instances or the number of threads dedicated to a particular file type, with the goal of minimizing the wait time of the publications in the queue. The RabbitMQ management tool can be enabled by running the following command:

```
rabbitmq-plugins enable rabbitmq_management
```

Note: Node restart is not required after activation of the plug-in.

After the RabbitMQ management user interface is enabled, it can be accessed from a web browser using the following URL: `http://{node-hostname}:15672`

The following screen shot shows an example of the queue messages displayed:



Queues

All queues (10)

Pagination

Page 1 of 1 - Filter: □ Regex ?

Overview					Messages			Message rates		
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/	opentext-ajira:staging:doc	classic	D Pri	idle	0	0	0			
/	opentext-ajira:staging:drw	classic	D Pri	idle	0	0	0			
/	opentext-ajira:staging:expired-publications	classic	D Pri	idle	0	0	0			
/	opentext-ajira:staging:pdf	classic	D Pri	idle	0	0	0			
/	opentext-ajira:type:doc	classic	D Pri	idle	0	0	0	0.00/s	0.00/s	0.00/s
/	opentext-ajira:type:drw	classic	D Pri	idle	0	0	0	0.00/s	0.00/s	0.00/s
/	opentext-ajira:type:expired-publications	classic	D Pri	idle	0	0	0	0.00/s	0.00/s	0.00/s
/	opentext-ajira:type:pdf	classic	D Pri	idle	0	0	0	0.00/s	0.00/s	0.00/s

By default, each Publisher instance processes three jobs concurrently per queue. The number of concurrent jobs can be modified by updating the queue properties in the `publisher.properties` configuration file and restarting the Publisher Service instance. Increasing this value allows the Publisher to pick up more jobs from the queue. However, higher concurrency increases the risk that available resources might be insufficient to process all concurrent jobs, potentially resulting in longer completion times for all jobs. When updating this property, it is important to find a careful balance between available computational and infrastructure resources (CPU, memory, disk I\O) and resource allocation.

For hyperscaler deployment environments, such as Kubernetes, the general recommendation is to increase the number of Publisher instances as much as possible instead of updating the queue priorities as just described.

To ensure consistent performance in publication processing, it is highly recommended to monitor the growth of publication-related RabbitMQ queues. RabbitMQ provides comprehensive monitoring capabilities through various channels, such as New Relic, Prometheus, and the RabbitMQ management console. When a defined threshold of messages in a queue has been reached, action can be taken to either increase the number of Publisher instances or adjust the number of queue job counts (DRW, PDF, DOC) to reduce the number of jobs in the queue.