

Perceptive Content Rendering and Conversion Service

Installation and Setup Guide

Version: 1.x

Written by: Documentation Team, R&D

Date: April 2024

Documentation Notice

Information in this document is subject to change without notice. The software described in this document is furnished only under a separate license agreement and may only be used or copied according to the terms of such agreement. It is against the law to copy the software except as specifically allowed in the license agreement. This document or accompanying materials may contain certain information which is confidential information of Hyland Software, Inc. and its affiliates, and which may be subject to the confidentiality provisions agreed to by you.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright law, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Hyland Software, Inc. or one of its affiliates.

Hyland, HXP, OnBase, Alfresco, Nuxeo, and product names are registered and/or unregistered trademarks of Hyland Software, Inc. and its affiliates in the United States and other countries. All other trademarks, service marks, trade names and products of other companies are the property of their respective owners.

© 2024 Hyland Software, Inc. and its affiliates.

The information in this document may contain technology as defined by the Export Administration Regulations (EAR) and could be subject to the Export Control Laws of the U.S. Government including for the EAR and trade and economic sanctions maintained by the Office of Foreign Assets Control as well as the export controls laws of your entity's local jurisdiction. Transfer of such technology by any means to a foreign person, whether in the United States or abroad, could require export licensing or other approval from the U.S. Government and the export authority of your entity's jurisdiction. You are responsible for ensuring that you have any required approvals prior to export.

Table of Contents

Documentation Notice	2
Overview	5
Download	5
Prerequisites	5
Operating systems.....	5
Private and public key creation	5
Token authentication	6
Install	7
Configure	7
Titan 4.0 or higher	7
Titan 3.0 or lower.....	9
Validate	10
Appendix A: appsettings.json file	11
Kestrel.....	11
Swagger.....	11
Hyland.Logging.....	11
Authentication	12
HostOptions	12
Identity	12
ApiKey	13
RateLimiter	13
FileCache.....	13
DocumentSessionManager	13
DocFiltersSessionFactory.....	13
DocFiltersRedactionFactory	14
ThumbnailCache	15
Metrics	15
TempPath	15
Appendix B: Hyland Application Settings Utility (HASU)	16
Register	16
<i>Example</i>	16
Write	16
<i>Example</i>	17

Read	17
<i>Example</i>	17

Overview

Perceptive Content Rendering and Conversion Service is a RESTful API which leverages Document Filters to provide conversion, extraction, and processing for hundreds of file types.

Download

To download Perceptive product installation files, complete the following steps.

1. Go to the Hyland Community site.
2. From the menu, click **Support** and then under **Software Downloads** select **Perceptive Downloads**.
3. Find and download the installer file corresponding to the version to be installed.

Note New and updated documentation and help topics are regularly published to the documentation website at docs.hyland.com.

Prerequisites

- When using Titan 4.0 or higher, the Hyland.Application.Settings.Utility (HASU) software is required to encrypt the **Identity:Inspection:ClientToken** setting. HASU can be obtained from the **Hyland Community Site > Support > Software Downloads**.

Operating systems

- We recommend upgrading to the latest operating system patch level for one of the following supported operating system versions:
 - Microsoft Windows Server 2016, 2019, or 2022

Private and public key creation

If you are using Titan 4.0 or higher, the Perceptive Content Rendering and Conversion Service uses a bearer token created and signed by Perceptive Content when authenticating with Integration Server. That bearer token requires that a key pair has been generated and imported into Perceptive Content.

You should generate a key pair using your organizations best practices. The key must use either the RSA or EC algorithm with a length of no less than 1024.

Note The service returns unauthorized errors if configured incorrectly. Refer to the service logs to determine any issues with setup.

To generate a new Java Keystore and RSA key pair, complete the following steps.

1. Open a command window and navigate to the bin directory of your JRE installation.
2. To generate a new keystore and key pair, run the following command.

```
keytool -genkeypair -alias <alias> -keyalg RSA -keystore <path-to-keystore> -  
keysize <keysize> -storetype PKCS12
```

Example

```
keytool -genkeypair -alias pcrs -keyalg RSA -keystore c:\cert\keystore -keysize  
2048 -storetype PKCS12
```

Setting	Options	Description
-alias	Any valid string.	Specifies the identifier associated with the key pair to be generated in the created keystore.
-keyalg	RSA	Specifies the algorithm used to generate the key pair.
-keystore	Any valid path.	Specifies the path where the keystore is created. Note Only JKS Keystores are supported.
-keysize	Any valid number, minimum 1024.	Specifies the length of the key to be generated.

3. To export the private key from the keystore, run the following command.

```
openssl pkcs12 -in <path-to-keystore> -nocerts -out <path-to-private-key>
```

Example

```
openssl pkcs12 -in c:\cert\keystore -nocerts -out c:\cert\pkrscert.pem
```

Setting	Options	Description
-in	Any valid path.	Specifies the keystore path created in the previous step.
-out	Any valid path.	Specifies the path where the private key will be exported.

4. To export the certificate, run the following command.

```
openssl pkcs12 -in <path-to-keystore> -nokeys -out <path-to-cert>
```

Example

```
openssl pkcs12 --in c:\cert\keystore -nokeys -out c:\cert\pkrscert.crt
```

Setting	Options	Description
-in	Any valid path.	Specifies the certificate path exported in the previous step.
-out	Any valid path.	Specifies the path where the public certificate will be exported.

5. Use the created key pair to configure Token Authentication in Perceptive Content.

Token authentication

If you are using Titan 4.0 or higher, a token signing key is required to configure Perceptive Content for Integration Server bearer token authentication. To configure a token signing key, complete the following steps using the key pair generated above.

1. In a text editor, open the **inow.ini** configuration file.
2. If you are using an RSA key, ensure that the `encryption.asymmetric.min.key.strength` and `encryption.asymmetric.max.key.strength` settings under **[Network]** encompass the size of the generated key used for token signing.
3. Under the **[Logon Control]** section, set `token.signing.key.path` to the file path of the private key used to sign tokens. If the private key is encrypted, set `token.signing.key.password` to the private key password. The token signing key must be in Base64 encoded DER (PEM) format.
4. In the same section, set the `token.signing.algorithm` to the algorithm used when signing tokens. Refer to inow.ini configuration file documentation for a list of supported values.

Example:

```
[Logon Control]
token.signing.key.path=c:\cert\pcrcscert.pem
token.signing.key.password=password
token.signing.key.password.encrypted=
token.signing.algorithm= RS256
```

5. Save the file and close the text editor.
6. Import the certificate or public key associated with the configured private key using either the `import-cert` or the `import-public-key` **intool** commands, specifying `token-auth` for the SSO type. Refer to the **intool** documentation for more information.

Example

```
intool.exe --cmd import-cert --file c:\cert\pcrcscert.crt --type token-auth
```

Install

To install Perceptive Content Rendering and Conversion Service, complete the following steps.

1. In the **Windows Explorer**, right-click the installer and select **Run as Administrator**. The system opens the **Perceptive Content Rendering and Conversion Service** wizard.
2. On the **End User License Agreement (EULA)** page, if you want to change the default location of the installation, click **Options** and then complete the following substeps.
 1. On the **Setup Options** page, in the **Install location** field, type or browse to the location of where you want to install the service.
 2. Click **OK**.
3. On the **License Agreement** page, read the agreement, select **I agree to the license terms and conditions**, and then click **Install**.
4. On the **Completed Perceptive Content Rendering and Conversion Service Setup Wizard** page, click **Finish**.

Configure

Titan 4.0 or higher

To configure Perceptive Content Rendering and Conversion Service for use with **Titan 4.0 or higher**, complete the following steps.

1. For a first install, navigate to the **Program Files\Hyland\Perceptive Content Rendering and Conversion Service\samples** folder.
2. Copy the **appsettings.sample.json** configuration file to the default root install directory, **Program Files\Hyland\Perceptive Content Rendering and Conversion Service**, and rename the file to **appsettings.json**.
3. Open the **appsettings.json** configuration file in a text editor.
4. In the **Kestrel.Certificates.Default** section, complete one of the following actions to update the settings to load a certificate for https connections.
 - For pfx certificate files, complete the following substeps.

1. Update the **Path** setting to be the location of the pfx file.
 2. Update the **Password** setting to be the password for the pfx file.
- To use certificates of other types or to use the certificate trust store see the [Configure endpoints for the ASP.NET Core Kestrel web server](#) topic on the Microsoft website.

```
"Kestrel": {
  "Endpoints": {
    "Https": {
      "Url": "https://*:5001"
    }
  },
  "Certificates": {
    "Default": {
      "Path": "C:\\sample.pfx",
      "Password": "5@mpl3P@55w0rd"
    }
  }
},
```

5. In the **Identity** section update the settings to validate the bearer token is from your identity provider
 1. Update the **Identity:ScopeName** settings to the scope required for the bearer token. If you do not want to require a scope, remove the **Identity:ScopeName** field completely from the configuration.

Note If you want to require a scope and are using Titan, the **Identity:ScopeName** setting must match one of the **scopes** defined under the **renderingConversionService** section in the Titan **app.config.json** configuration file and one of the scopes defined in **psw.bearer.token.allowed.scopes** under the **Perceptive Token Management** section in the Perceptive Content Server **inserver.ini**.

2. Update the **Identity:Introspection:Endpoint** setting to the Introspection Authority. If this is not set, it will use the **Identity:Authority** in the Open ID Connect discovery document.

Note When using Titan, the **Identity:Introspection:Endpoint** setting is required and is expected to be the Integration Server introspection endpoint.

3. Update either the **Identity:Introspection:ClientToken** setting or both the **Identity:Introspection:ClientId** and **Identity:Introspection:ClientSecret** settings.

Note When using Titan, the **Identity:Introspection:ClientToken** setting is required. For the Integration Server introspection endpoint the ClientToken should be set to the bearer token generated using the steps in the Manage Content help topic titled [Acquire a Perceptive introspection bearer token](#).

For more information, see [Private and public key creation](#) and [Token authentication](#).

```
"Identity": {
  "Authority": null,
  "JwtValidationClockSkewMinutes": 0,
  "RequireHttpsMetadata": true,
  "ScopeName": "custom-scope",
  "Introspection": {
    "TokenFormat": "Any",
    "Endpoint":
"https://sample.onbase.com/integrationserver/v1/authorization/perceptivetoken/i
ntrospect",
    "ClientToken": "some_client_token"
  }
},
```

```
    }
  },
```

Example

The following is an example of running HASU to encrypt the ClientToken setting. For more information, see [Appendix B: Hyland Application Settings Utility \(HASU\)](#).

```
Hyland.Application.Settings.Utility.exe write -a "C:\Program
Files\Hyland\Perceptive Content Rendering and Conversion Service" --file
appsettings.json -p Identity:Introspection:ClientToken -i
```

6. Update the **HostOptions.AllowedOrigins** setting to a list of web clients to allow to connect to the service.

```
"HostOptions" : {
  "AllowedOrigins" : [
    "https://titan.onbase.net",
    "https://titan2.onbase.net"
  ],
  "MultipartBodyLengthLimit": 134217728
},
```

7. Save the **appsettings.json** configuration file and close the text editor.
8. Restart the **Perceptive Content Rendering and Conversion Service** to make the changes effective.

Titan 3.0 or lower

To configure Perceptive Content Rendering and Conversion Service for use with **Titan 3.0 or lower**, complete the following eight steps.

1. For a first install, navigate to the **Program Files\Hyland\Perceptive Content Rendering and Conversion Service\samples** folder.
2. Copy the **appsettings.sample.json** configuration file to the default root install directory, **Program Files\Hyland\Perceptive Content Rendering and Conversion Service**, and rename the file to **appsettings.json**.
3. Open the **appsettings.json** configuration file in a text editor.
4. In the **Kestrel.Certificates.Default** section, complete one of the following actions to update the settings to load a certificate for https connections.
 - For pfx certificate files, complete the following substeps.
 1. Update the **Path** setting to be the location of the pfx file.
 2. Update the **Password** setting to be the password for the pfx file.
 - To use certificates of other types or to use the certificate trust store see the [Configure endpoints for the ASP.NET Core Kestrel web server](#) topic on the Microsoft website.

```

"Kestrel": {
  "Endpoints": {
    "Https": {
      "Url": "https://*:5001"
    }
  },
  "Certificates": {
    "Default": {
      "Path": "C:\\sample.pfx",
      "Password": "5@mpl3P@55w0rd"
    }
  }
},

```

5. In the **Identity** section update the settings to validate the bearer token is from your identity provider
 1. Update the **Authority** setting to the URL of your identity provider. Note that this setting is the same value configured for the **issuer** setting under the **authConfig** section in the Titan **app.config.json** configuration file.
 2. Update the **Identity:ScopeName** settings to the scope required for the bearer token. If you do not want to require a scope, remove the **Identity:ScopeName** field completely from the configuration.

Note If you want to require a scope and are using Titan, the **Identity:ScopeName** setting must match one of the **scopes** defined under the **authConfig** section in the Titan **app.config.json** configuration file.

3. Remove the entire **Identity:Introspection** section.

```

"Identity": {
  "Authority": " https://IDP-hostname/identityprovider",
  "JwtValidationClockSkewMinutes": 0,
  "RequireHttpsMetadata": true,
  "ScopeName": "custom-scope"
},

```

6. Update the **HostOptions.AllowedOrigins** setting to a list of web clients to allow to connect to the service.

```

"HostOptions" : {
  "AllowedOrigins" : [
    "https://titan.onbase.net",
    "https://titan2.onbase.net"
  ],
  "MultipartBodyLengthLimit": 134217728
},

```

7. Save the **appsettings.json** configuration file and close the text editor.
8. Restart the **Perceptive Content Rendering and Conversion Service** to make the changes effective.

Validate

To verify the Perceptive Content Rendering and Conversion Service is running successfully, complete the following action.

- Go to <https://<hostname>:5001/healthcheck>. If the status is healthy, the system should display the following status message.

```
{
  "healthy": {
    "Private Memory Size": "OK. 4294967296",
    "Working Set": "OK. 4294967296 bytes",
  },
  "status": "Healthy"
}
```

Appendix A: appsettings.json file

You can configure the Perceptive Content Rendering and Conversion Service by updating the appsettings.json file. The following is a list of the sections available in the appsettings.json file.

Kestrel

The **Kestrel** section specifies the values for the Kestrel webserver. The Kestrel webserver is used when running as a standalone application or as a windows service. You can configure the HTTP endpoint here. The default is <https://0.0.0.0/5001>. The 0.0.0.0 syntax specifies binding to all address on the host. For more information, see the [Configure endpoints for the ASP.NET Core Kestrel web server](#) topic on the Microsoft website. Depending on the types of certificates being used, refer to this link for the values needed for your configuration.

Note In some applications, the size of the page file that can be rendered is limited by the **MaxRequestBodySize** setting. The default is 28.6 MB. If you work with files larger than the default, increase the value of this setting to allow those larger page files to render.

```
"Kestrel": {
  "Endpoints": {
    "Https": {
      "Url": "https://*:5001"
    }
  },
  "Limits": {
    "MaxRequestBodySize": <size in bytes>
  },
  "Certificates": {
    "Default": {
      "Path": "",
      "Password": ""
    }
  }
},
```

Swagger

The **Swagger** section specifies whether the Swagger service is enabled or disabled.

```
"Swagger": {
  "Enabled": false
},
```

Hyland.Logging

The **Hyland.Logging** section specifies the logging configuration for Perceptive Content Rendering and Conversion Service.

```

"Hyland.Logging": {
  "Routes": {
    "Console-Full": {
      "DisableIPAddressMasking": "true",
      "minimum-level": "Trace",
      "maximum-level": "Critical",
      "Console": "value"
    },
    "Http": {
      "Http": null,
      "DisableIPAddressMasking": "true",
      "minimum-level": "Trace",
      "maximum-level": "Critical"
    }
  }
},

```

Authentication

The **Authentication** section specifies the underlying authentication provider that the system uses to authenticate credentials that are passed to the Perceptive Content Rendering and Conversion Service.

```
"Authentication" : [ "Bearer" ],
```

Note Self-contained access tokens are required when using Bearer authentication. In Hyland IDP, set the Access Token type to JWT.

HostOptions

The **AllowedOrigin** section specifies an accepted origin to make CORS (Cross Origin Resource Sharing) requests. For more information, see the [Enable Cross-Origin Requests \(CORS\) in ASP.NET Core](#) topic on the Microsoft website.

The **MultipartBodyLengthLimit** value is the max number of bytes for multipart form data used on file upload.

Note In some applications, the size of the page files that can be rendered is limited by the **MultipartBodyLengthLimit** setting. The default is 128 MB. If you work with files larger than the default, increase the value of this setting to allow those larger page files to render.

```

"HostOptions" : {
  "AllowedOrigins" : [
  ],
  "MultipartBodyLengthLimit": 134217728
},

```

Identity

The **Identity** section specifies the configuration options for validating Bearer authentication.

```

"Identity": {
  "Authority": "https://sample.onbase.com/identityprovider",
  "JwtValidationClockSkewMinutes": 0,
  "RequireHttpsMetadata": true,
  "ScopeName": "custom-scope",
  "Introspection": {
    "TokenFormat": "Any",

```

```

        "Endpoint":
"https://sample.onbase.com/identityprovider/connect/introspectintegrationserver/v1/aut
horization/perceptivetoken/introspect ",
        "ClientToken": "some_client_token"
    },

```

ApiKey

The **ApiKey** section specifies the set of keys to allow when the Authentication configuration option contains ApiKey.

```
"ApiKey" : [],
```

RateLimiter

The **RateLimiter** section specifies the configuration options that determine rate limits for the Conversion endpoint for the Perceptive Content Rendering and Conversion Service. These options do not impact the Session endpoint.

```

"RateLimiter" : {
    "MaxConnections" : 20,
    "TimeoutMilliseconds" : 30000
},

```

FileCache

The **FileCache** section specifies the configuration options for implementing the file cache for document sessions.

```

"FileCache" : {
    "RootPath" : null,
    "DefaultTimeout" : "0:00:00:30.0",
    "RootPathSlidingExpiration" : "1:00:00:00.0",
    "RootPathExpirationScanFrequency" : "0:01:00:00.0",
},

```

DocumentSessionManager

The **DocumentSessionManager** section specifies the configuration options for controlling memory caching.

```

"DocumentSessionManager" : {
    "ObjectCache" : {
        "MaxObjects" : 32,
        "SlidingExpiration" : "0:00:02:00.0",
        "CompressFrequency" : "0:00:02:00.0",
    }
},

```

DocFiltersSessionFactory

Note If you need to update this section of the configuration file, contact your system administrator or support specialist.

The **DocFiltersSessionFactory** section specifies the options for controlling **DocumentFilters.Extractor** and **DocumentFilters.Page** objects along with their memory caching.

The following lists OpenType supported values.

- BodyOnly
- MetaOnly
- BodyAndMeta
- FormatText
- FormatHTML
- FormatXML
- FormatHTMLIfYouCan
- FormatImage
- FormatHTMLNoFallback

```
"DocFiltersSessionFactory" : {
  "MemoryCacheOptions" : {
    "ExpirationScanFrequency" : "0:00:01:00.0",
    "SizeLimit" : 50,
    "CompactionPercentage" : 0.05
  },
  "MemoryCacheEntryOptions" : {
    "AbsoluteExpiration" : null,
    "AbsoluteExpirationRelativeToNow" : null,
    "SlidingExpiration" : "00:00:05:00.0",
    "Priority" : "Normal",
    "Size" : 1
  },
  "DocFiltersOptions" : {
    "DocFiltersOpenType" : "FormatImage",
    "DocFiltersAdditionalOpenOptions" : {
      "LIMITS_PAGE_COUNT" : "2000"
    },
    "DocumentSessionOptions" : {
      "ExtractPageElementWordLimit" : 500000
    }
  }
},
```

DocFiltersRedactionFactory

Note If you need to update this section of the configuration file, contact your system administrator or support specialist.

The **DocFiltersRedactionFactory** specifies how the system stores document sessions with redactions. Redacted documents are rendered locally to a new document of **DocumentCanvasType** with pages containing redactions being rendered as **PageRasterCanvasType** within the larger document canvas. The result of this render is used to generate the document session id and all future document session functionality.

The following lists the **CanvasType** supported values.

- PNG
- JPG
- PDF
- TIF

- BMP
- XML
- HTML
- PBM
- PGM
- PPM
- WEBP
- XPS
- SVG
- EPS
- PS

```
"DocFiltersRedactionFactory" : {
  "DocumentCanvasType" : "PDF",
  "PageRasterCanvasType" : "PNG",
  "DocFiltersAdditionalOpenOptions" : {
    "LIMITS_PAGE_COUNT" : "2000"
  }
},
```

ThumbnailCache

The **ThumbnailCache** section specifies the thumbnail cache within Perceptive Content Rendering and Conversion Service.

```
"ThumbnailCache" : {
  "RootPath" : null
},
```

Metrics

The **Metrics** section specifies the metric gathering options for Perceptive Content Rendering and Conversion Service, and exposes options related to this gathering. The memory thresholds indicate upper limit thresholds. If the consumed memory of the service is at or above the configured thresholds, the metrics reporting endpoints will indicate that the service is in degraded, or unhealthy states.

```
"Metrics": {
  "Enabled" : true,
  "PhysicalMemoryThreshold": "4GB",
  "PrivateMemoryThreshold": "4GB"
},
"OperationLimits" : {
  "SessionFind" : {
    "TimeInMs" : "5000"
  }
},
```

TempPath

The **TempPath** section specifies the path used for temporary files. For example, you may want to use temporary files in Perceptive Content Rendering and Conversion Service when downloading a file from a

consumer-provided URI. The service writes the downloaded data to a temporary file when a user uploads an HTTP form file, writes the uploaded data to a temporary file when a user requests a subfile from a session, and then writes the extracted subfile data to a temporary file

```
"TempPath": null
}
```

Appendix B: Hyland Application Settings Utility (HASU)

This section gives a brief overview of how to use HASU, a command line utility, to register certificates to the required certificate store and write values to an existing property within a configuration file.

Register

The **register** command registers a valid x509Certificate2 into the cert store in CurrentUser:My store location for future encryption and decryption. This is required for Linux based environments.

The following table list the available actions for register.

Action	Description
-p --password	Required. Specifies the password that protects the certificate.
--f --filePath	Required. Specifies the full filepath to the PFX certificate.
--verbose	Sets minimum LogLevel to Trace. The default is Error.

Example

```
./Hyland.Application.Settings.Utility register --filePath C:\cert\sample-
certificate.pfx --password YourPassword --verbose
```

Write

The **write** command stores the value to an existing property within the configuration file.

The following table lists the available actions for write.

Action	Description
-p, --property	Required. Property of the config file. Nested objects should be separated by a colon ':'.
-a, --applicationRoot	Required. Specifies the path to the application root.
--file	Required. Specifies the relative filepath to the JSON configuration from the application root.
-v, --value	Required. Specifies the value to protect.
-i, --inline	Required. Encrypts the value of the provided property in the provided file.
-r, --recursive	Required. Encrypt all of the values under the provided property in the provided file.

<code>--verbose</code>	Sets minimum LogLevel to Trace. The default is Error.
<code>-f, --force</code>	Forces creation of value if not present. The default is false.
<code>--encrypt</code>	Boolean flag to determine if the value should be encrypted before being stored. The default is false. If needed, you can read the encrypted settings stored in the configuration file.
<code>-t, --thumbprint</code>	Required for Linux. Specifies the certificate thumbprint used to hash the encryption key.

Example

```
./Hyland.Application.Settings.Utility write -a "C:\Program Files\Hyland\Perceptive Content Rendering and Conversion Service" --file appsettings.json -p Identity:Inspection:ClientToken -i --thumbprint 03D724DD2666B9D858CAB84808372BAE82F89A36 --encrypt
```

Read

The **read** command displays the property value in the console output.

The following table lists the available actions for read.

Action	Description
<code>-p, --property</code>	Required. Property of the config file. Nested objects should be separated by a colon ':'.
<code>-a, --applicationRoot</code>	Required. Specifies the path to the application root.
<code>--file</code>	Required. Specifies the relative filepath to the JSON configuration from the application root.
<code>--verbose</code>	Sets minimum LogLevel to Trace. The default is Error.
<code>--decrypt</code>	Boolean flag to determine if the value should be decrypted before being retrieved. The default is false.

Example

```
./Hyland.Application.Settings.Utility read -a "C:\Program Files\Hyland\Perceptive Content Rendering and Conversion Service" --file appsettings.json -p Identity:Inspection:ClientToken
```