dox42 Server V3.5.1-5.0

Documentation

Summary

This document explains the service interface as well as the configuration for the dox42 server.

Content

Document details

Version: dox42 Server V3.5.1-5.0

Authors: Christian Bauer, Lisa Pulsinger

Date: 05 March 2025

Service interface

The dox42 server is accessible via a SOAP and a REST WebService interface.

Operation: GenerateDocument/GenerateSlides

The operation can be operated with the **following parameters:**

- **DocTemplate** (String)
  path to the document template (e.g. C:\doctemplates\solar_subsidy.docx, C:\doctemplates\salesreport.pptx)
- **InputParams** (Key-Value String/String)
  list of the input parameters and values:
  - **ParamName**: name of the parameter (not case-sensitive)
  - **Value**: value of the parameter
- **PostGenActions**
  list of PostGenerateActions to process after generating the document:
  - **ActionName:** name of the action (e.g. SaveAction), not case-sensitive
  - **Params:** parameter of the action (like InputParams)

Operation: GenerateSpreadSheet

The operation can be operated with the **following parameters:**

- **DataMap** (String)
  path to the data map (e.g. C:\doctemplates\salesreport.dm)
- **InputParams** (Key-Value String/String)
  list of the input parameters and values:
  - **ParamName**: name of the parameter (not case-sensitive)
  - **Value**: value of the parameter
- **PostGenActions**
  list of PostGenerateActions to process after generating the document:
  - **ActionName:** name of the action (e.g. SaveAction), not case-sensitive
  - **Params:** parameter of the action (like InputParams)

The paramameters of all individual OutputActions are described in this document.

SOAP Interface

URL: http://www.YourDomain.com/dox42service.asmx

WSDL: http://www.YourDomain.com/dox42service.asmx?WSDL

REST Interface

URL: http://www.YourDomain.com/dox42RestService.ashx

The parameters and output actions for the REST and the SOAP interface are identical. The REST Interface uses the following URL-parameter syntax. Since version 3.6.3 you can also pass all parameter via POST:

Parametername	Erfordernis	Auswirkung
Operation	mandatory GenerateDocument | GenerateSpreadSheet | GenerateSlides	The operation to call
DocTemplate	mandatory if Operation=GenerateDocument or GenerateSlides	Path of the document template
DataMap	mandatory if Operation= GenerateSpreadSheet	Path of the data map
InputParam. <ParameterName>	For each input parameter	Values for input parameters, e.g.: InputParam.CustomerID=42
<OutputAction>. <ParameterName>	For each parameter of each OutputAction	Values for OutputAction-Parameters, z.B.: ReturnAction.Format=pdf
ReturnAction.disp	optional Inline | attachment	Controls the how the web browser handles the returned document
ReturnAction.Mode	optional File (Standard) | Text	Use Text to render the content of the generated document directly in the browser as text. This text may be HTML.
ReturnAction.fileName	optional, only if ReturnAction.disp = attachment	Name of the returned document
RedirectURL	Optional, only without ReturnAction	Redirect to this page after generation

ReturnSavedFileInfo	Optional (since 4.5)	If true, REST Calls return information about saved files as JSON.

Caution: Use URL-Encoding! (http://www.w3schools.com/tags/ref_urlencode.asp):
„Customer.Name=Müller“ => „ Customer.Name=M%C3%BCller “

Use the syntax <%datasource.datafield%> as a place holder for dynamic values in your calls.

Sample for a REST call:

http://www.YourDomain.com/dox42RestService.ashx?
Operation=GenerateDocument&
DocTemplate=%7e%5cTemplates%5cSales_Report%5cSales_Report.doc &
InputParam.SelectEmployee=5&InputParam.SelectYear=2012&
ReturnAction.format=pdf&ReturnAction.fileName=SalesReport_<%Employees.Name%>.pdf&Return
Action.disp=attachment

Your dox42 calls can easily be configured with our dox42 Server Designer application, which you can download from our website.

HTTP Request-Body

In case you need to transfer lots of data to the dox42 Server you better do that as POST parameters or in the Request-Body (since version 4.0). To extract the Request-Body you need to define an InputParameter of the name „RequestBody“ in your DataMap. The data from the HTTP Request-Body will be stored to this InputParameter and can be processed using a XML/JSON data source for example.

Service-Header

A service header secures the dox42 SOAP service. This header contains the user name and the password. Use web.config to configure the utilisation of the service header.

Azure Active Directory Header

The dox42 Service can process an Authorization Header, containing an Azure Active Directory Token (since version 4.0).

Javascript Example:

```
xhttp.setRequestHeader("Authorization", 'Bearer ' + token);
```

HTTP Metadata Header and Base64 Metadata Header

Since V 4.6. you can send a custom string containing datafields in a dedicated dox42 HTTP Header “dox42MetaData”. Here is an example of such a string, the header will be returend with datafields replaced:

{"sender": "<%User.mail%>"

"receiver": "<%Customer.mail%>"

"subject": "Here is your document. <%quote.number%>"}

Since Version 4.6.4. you can also use “dox42MetaDataBeaderBase64” to send base64 encoded metadata in your header.

OutputActions

The product dox42 server contains the following OutputActions:

- **SaveAction:** Saves the generated document to a file folder
- **SharepointAction:** Stores the generated document to a Sharepoint document library
- **EmailAction:** Sends the generated document via email
- **ReturnAction:** Returns the generated document via the WebServices interface
- **PrintAction:** Print generated document
- **dox42LiveAction:** Used for calls with the product dox42 Live
- **AzureBlobStorageAction:** Stores the generated document on an Azure Blob Storage
- **DropBoxAction:** Stores the generated document on Dropbox. This dox42 Output Acion is described in the dox42 Dropbox Action documentation.
- **DocusignAcion:** Sends the generated document to Docusign for electronic signature. This dox42 Output action is described in the dox42 Docusign documentation
- **SQLAction:** Stores the generated document in a SQL database. This output action is described in the dox42 SQL Output Action.
- **CustomOutputAction:** You can implement and use your own dox42 OutputAction. Custom Output Actions have to be registered in the web.config (analog Custom Data Sources) and use the following interface: `dox42CustomOutputActionKit. ICustomOutputAction`

All additional documenations mentioned above can be found here:
https://www.dox42.com/Resources?filter[]=.dokumentation&

Since version 4.6. the optional parameter “Condition” is available for your dox42 Output Actions. If false, the selected output action will not be performed.

In the following, the parameters of the OutputActions are described. None of the parameter names are case-sensitive. All parameter names may contain data fields (for exceptions see table). Example: FileName = “Application_<%Citizen.Name%>.pdf”

SaveAction

Parameter name	Requirement	Effect
FileName	mandatory	Defines the name of the document, which it will be saved as. e.g. C:\Documents\SalesReport.pdf The extension will define the file format (.docx, .xlsx, .pptx, .pdf, .html, …). See all output formats below.
SpecialFormat (dox42ForDocuments only)	optional	When generating a PDF, the format may be specified: PDFA1b and PDFForm. For PDFForm Form fields are preserved in the PDF. For HTML, you may use these options: **HTMLCode** which will cause the text of the dox42 template to be interpreted in HTML code. **HTMLZIP** The HTML -file along with the contained images will be packaged in a ZIP-file. (since version 3.6.3). For text files (*.txt) you can set an alternate extension. That is useful for generating meta data files for archiving-systems. Example: FileName = Order_<%Cust.OderNo%>.txt SpecialFormat = jpl Result: Order_42.jpl (since version 3.5)
metadatafilepath	optional	Generates an additional file with meta data information. E.g.: C:/dox42Server/generateddocuments/metadata.xml
metadatafilecontent	optional	Defines the content that is written to the metadata file.
Sign	optional	Value: true The format PDF may bear a digital signature (dox42ForDocuments only). Certificate and corresponding password have to be defined in the web.config.
Condition	Optional (since V. 4.6)	If “false”, the SaveAction will not be performed

Possible Output Formats:

Word: doc, docx, docm, dot, dotx, pdf, html, txt, jpg, png, tiff, bmp, rtf, odt
Excel: xls, xlsx, xlsm, xlt, xltx, xltm, xlsb, txt, csv, html, pdf, xml, tiff, ods
PowerPoint: pptx, pps, pdf, tiff

SharepointAction

Parameter name	Requirement	Effect
Site	mandatory	The Sharepoint site e.g. http://mysite.com
Library	mandatory	The Sharepoint document library root folder name
FileName	mandatory	Defines the name of the document, which it will be saved as. e.g. SalesReport.pdf The extension will define the file format (.docx, .xlsx, .pptx, .pdf, .html, …).
Folder	optional	Saves the document to the specified folder.
Rootfolder	Optional (since Version 4.4.1.7)	Define the internal name of your SharePoint library here.
SpecialFormat (dox42ForDocuments only)	optional	When generating a PDF, the format may be specified: PDFA1b and PDFForm. For PDFForm Form fields are preserved in the PDF. For HTML, you may use the option: HTMLCode which will cause the text of the generated document to be interpreted in HTML code.
Sign	optional	Value: true The format PDF may bear a digital signature (dox42ForDocuments only). Certificate and corresponding password have to be defined in the web.config.
Field1.Name **Field1.Value**	optional	Fields allow you to write values to columns in the Sharepoint document library. You can create multiple Fields: Field1, Field2, … Use the SharePoint internal field name to indentify the field. Tipp: You may find out the internal field name by opening the list in the dox42 SharePoint data source and check “Use Internal Field Names”.
ContentType	optional	Set this Content Type for the item

DocumentDataField	optional	If this field is assigned, the generated document will not be saved in the document library, but the document in the stated dox42 Datafield will be loaded (e.g. Data.MyDocument) and saved in the Sharepoint document library. The DocumentDataField can contain a Path, an URL or a binary Document.
CheckInMessage	optional	If this Field is assigned, the server will check in the document after saving with the stated message. If a document of the same title is already existing this document will be checked out automatically
Username	optional	If not provided, the service account of the dox42 Server will will be used for SharePoint authentication.
Password	optional	
SharePoint_online	optional	true/false Set this flag for SharePoint online (Office 365). Alternative to **Azure_AD**
SharePoint_FBA	optional	true/false Set this flag for SharePoint sites with forms based authentication.
Azure_AD	optional	true/false The Request must include an Authorization Header containing Azure Active Directory Token - alternative to **SharePoint_online**. (Since version 4.0).
retry_count	optional	A number >= 0 Defines that the SharePointAction should retry to save the document in case of error (e.g. timeout) and specifies the number of retries. Default = 0 (no retry). (Since version 4.1.3).
Condition	Optional (since V. 4.6)	If “false”, the SharePointAction will not be performed

Set special SharePoint and Managed Metadata Fields

For a choice field simply set the value.

To set a multi value choice field use “;#” to separate the values, e.g.: “red;#green;#yellow”.

For a Lookup/Person/Group-field set the ID.

To set a multi value Lookup/Person/Group-field, you need to set the ID and the value like „ID;#Value“, e.g.: „3;#Hawaii;#9;#Vienna;#13;Melbourne“. Actually only the IDs are used to set the field, the values are ignored (but may not be omitted).

EmailAction

Parameter name	Requirement	Effect
Mode	optional **Possible Values:** TxtEMail HtmlEMail AttachmentTxtBody Attachment (default)	Allows to configure whether the generated document should be used as the body of a txt- or html-mail or attached to the email. Use TxtEMail and HtmlEMail only with GenerateDocument.
FileName	mandatory if mode = Attachment, otherwise ignored	Name of the attachment The extension will define the file format (.docx, .xlsx, .pptx, .pdf, .html, …).
SpecialFormat (dox42ForDocuments only)	optional	When generating a PDF, the format may be specified: PDFA1b and PDFForm. For PDFForm Form fields are preserved in the PDF. For HTML, you may use the option: HTMLCode which will cause the text of the generated document to be interpreted in HTML code.
Sign	optional	Value: true The format PDF may bear a digital signature (dox42ForDocuments only). Certificate and corresponding password have to be defined in the web.config.
Receiver	mandatory	Receivers email addresses Separate multiple addresses with a comma or semicolon.
ReceiverCC	optional	Carbon copy receivers
ReceiverBCC	optional	Blind carbon copy receivers
Sender	optional	Sender’s email address If no email address is specified, the address stated in the web.config will be used.

Parameter name	Requirement	Effect
MailServerUsername	optional	May be specified along with Sender if a different login for the mail server is required. Since Version 3.6.2.4
SenderPassWord	mandatory if Sender or MailServerUsername is specified, else: omit	Sender's password for his email account or MailServerUsername on the email server, configured in the web.config.
Header	mandatory	Email subject
Body	optional	Email text (mode = Attachment only)
HtmlBody	optional	Email text as HTML (mode = Attachment only)
BodyTemplate	optional	A dox42 Word Template to generate the email body. The template must use the same Datamap as the attachment. (mode = Attachment only) Since V 3.5.1 Mode = AttachmentTxtBody generates the body template in Txt format. (Since V. 4.4.1.5)
StaticAttachment1.Path **StaticAttachment1.Base64** **StaticAttachment1.Name**	optional (or use Base64) e.g. C:\Images\MyImage.jpg optional (or use Path) Base64 encoded attachment mandatory if Base64 is used e.g. MyImage.jpg	Allows sending for static attachments. It is possible to send any number of attachments, like: StaticAttachment2… StaticAttachment3… You may specify either the path of a static file, or pass the attachment base64 encoded.
MailServer	optional	Use this parameter if you want to use a different mail server than configured in your web.config.
MailServerPort	Optional Since version 3.4.3.10	Change the SMTP Mail Server Port. You may want to use the standard ports 25 or 587.

Parameter name	Requirement	Effect
SSL	Optional Since version 3.5	true or false
SaveMSGFile	Optional Since version 3.5.0.19	File path to save the sent mail as msg-file, e.g. C:\Windows\Temp\Order_<%Customer.ID%>.msg
SendMode	Optional Since Version 3.6	send or save **send**: Default, send mail immediately **save**: Don't send mail, but save the mail as draft to the path defined in SaveMSGFile. The parameter SaveMSGFile must be specified. **return**: Don't send mail, but return the mail as draft in the ReturnMessage. Since Version 3.6.3.
MailMethod	Optional Since Version 4.3.3	Use this parameter to choose a specific method to send your dox42 emails. Possible values: exchangeonline, aspose, mailkit, onmailkiterrortryaspose, onasposeerrortrymailkit **You need to use exchangeonline if you send emails via Microsoft Exchange Online.** **onasposeerrortrymailkit**: When the Aspose Mail Method throws an error, the mailkit method is used to send dox42 emails. When adding the MailMethod parameter you may also want to add the MailServerPort parameter, if you have not done so yet in your web.config.
AzureADAppId	Optional (Since Version 4.4.1.7)	Application ID of the app registration that has the necessary API permissions configured to send mails via Exchange Online (can be configured in the web.config/MAUI as well)
AzureADTenantId	Optional (Since Version 4.4.1.7)	Tenant ID in which the above app registration is configured to send mails via Exchange Online (can be configured in the web.config/MAUI as well)
AzureADClientSecret	Optional (Since Version 4.4.1.7)	Clientsecret that is associated with the App ID to send mails via Exchange Online (can be configured in the web.config/MAUI as well)

Parameter name	Requirement	Effect
ExchangeOnlineUrl	Optional (Since Version 4.4.1.7)	Comes with the default value https://outlook.office365.com/EWS/Exchange.asmx and can be left empty
Condition	Optional (since V. 4.6)	If “false”, the EmailAction will not be performed

To use an alias with an email address please use this syntax (since 3.6.2.5):
MyCompany Sales {sales@mycompany.com}

MyCompany Sales <sales@mycompany.com> , would work as well, but might cause an Error if used with the REST Service.

Exchange Online

To send emails via Microsoft Exchange Online, you need to register an application in Azure Active Directory. Further information on the Azure AD App registration can be found in our SharePoint documentation: https://www.dox42.com/Resources?filter[]=.dokumentation&

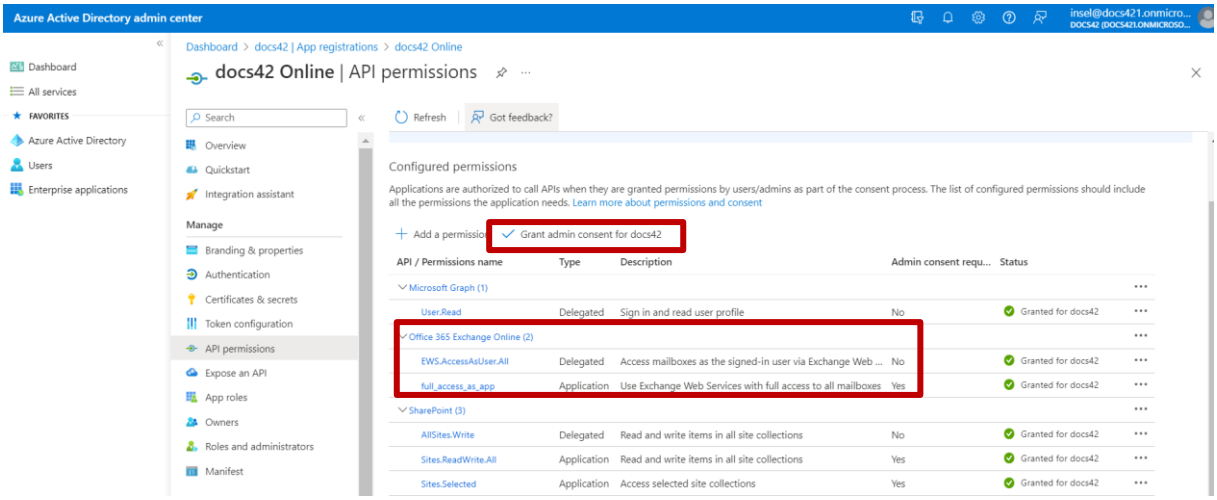
Furthermore, you need to add API permissions to this app registration that contain permissions to send Mails via Exchange online. This needs to be added via the Manifest:

Under the „requiredResourceAccess“ Section in the Manifest of the App Registration, you need to add a new JSON object with the following values:

You can copy and paste it from here:

```
{
    "resourceAppId": "00000002-0000-0ff1-ce00-000000000000",
    "resourceAccess": [
        {
            "id": "3b5f3d61-589b-4a3c-a359-5dd4b5ee5bd5",
            "type": "Scope"
        },
        {
            "id": "dc890d15-9560-4a4c-9b7f-a736ec74ec40",
            "type": "Role"
        }
    ]
}
```

Afterwards go to the API permissions section. The following additional permissions should appear: Click on “Grant admin Consent.”



Now you need to add a new Site to the AzureAD Section of your web.config or your dox42 Online MAUI. It needs to contain the App Id, Tenant ID and the following URL: https://outlook.office365.com/

When setting up email calls with Exchange online, do not forget to add the parameter mailmethod and the value “exchangeonline”.

ReturnAction

Parameter name	Requirement	Effect
Format	mandatory	File format (.docx, .xlsx, .pptx, .pdf, .html, emf (since v.4.6.2.)…).
Filename	Optional	Defines the file name of a document. E. g. Contract.pdf. To generate a dynamic file name, use the following syntax: Contract_<%employee.fullname%>.pdf
Mode	optional Possible Values: File (Standard) Text	Defines whether the generated Document should be returned as File (Mode=File) in ServiceResult.GeneratedDocs or as text (Mode=Text) in ServiceResult.ResultMessage.
SpecialFormat (dox42ForDocuments only)	optional	When generating a PDF, the format may be specified: PDFA1b and PDFForm. For PDFForm Form fields are preserved in the PDF. For HTML, you may use the option: HTMLCode which will cause the text of the generated document to be interpreted in HTML code. EMFZIP (since v. 4.6.2.)
Sign	optional	Value: true The format PDF may bear a digital signature (dox42ForDocuments only). Certificate and corresponding password have to be defined in the web.config.
Condition	optional (since V. 4.6)	If “false”, the ReturnAction will not be performed
Numberofcopies	optional (since v. 4.6.2.)	e.g. 1,2,3. Specifies how many copies should be generated within one file

PrintAction

Parameter name	Requirement	Effect
PrinterName	optional	The used printer
Collate	optional (true/false)	Sort?
Copies	optional	Amount of copies
Duplex	Optional (Vertical, Horizontal, Simplex)	Duplexprint
FromPage	optional	Print from pagenumber …
ToPage	optional	… to pagenumber
Condition	Optional (since V. 4.6)	If “false”, the PrintAction will not be performed

dox42LIVEAction

The dox42LIVEAction uses an output configuration from a dox42LIVE.config file to execute output actions. Please see the dox42 LIVE documentation to learn how to define your dox42LIVE.config.
To use a dox42LIVEAction from the dox42 SOAP or REST Service the dox42LIVE.config file needs to be stored in the same directory as dox42RestService.ashx and dox42Service.asmx , typically: C:\dox42Server.
Since version 4.1.3.

Parameter name	Requirement	Effect
LiveActionName	mandatory	The LiveActionName as defined in the dox42LIVE.config file.
Condition	Optional (since V. 4.6)	If “false”, the dox42LIVEAction will not be performed

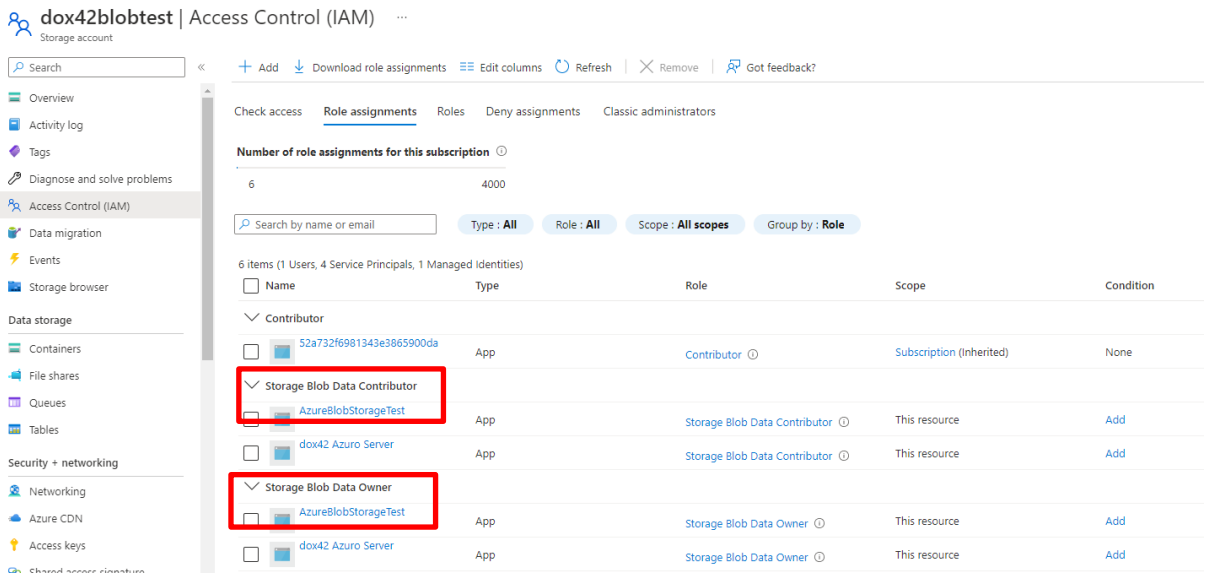
AzureBlobStorageAction

With the Azure BlobStorageAction you can store your generated documents in an Azure Blob storage container. (Since 4.3.1.)

You need to configure the following settings in Azure to use this action:

In your Azure AD App registration, add **delegated Azure Storage user impersonation** rights and grant admin consent. Further information about dox42 Azure AD App registration can be found in the dox42 SharePoint and D365 CE Documentation: https://www.dox42.com/Resources?filter[]=.dokumentation.SharePoint&filter[]=.dokumentation.CE-CRM&

Next, add your app registration as an Owner and as a Contributor to the Access Control Section of your Azure Storage account:



Finally, add https://storage.azure.com with the corresponding App Id, TenantID and Client Secret to the Azure AD Section in your web.config or your dox42 Online Maui. In your dox42 calls, use the following parameters:

Parameter name	Requirement	Effect
AzureBlobStorageAccountname	mandatory	The name of the Azure storage account.
AzureBlobStorageContainername	Mandatory	The name of the Azure storage container.
UploadToAzurePath	Mandatory	The file name (which can include directories) and extension of the uploaded file: Example: directoryA/document.docx
AzureBlobStorageContentType	Optional (since V. 5.0)	You can set a dynamic Content Type with this parameter.
Condition	Optional (since V. 4.6)	If “false”, the AzureBlobStorageAction will not be performed

ZUGFeRDAction

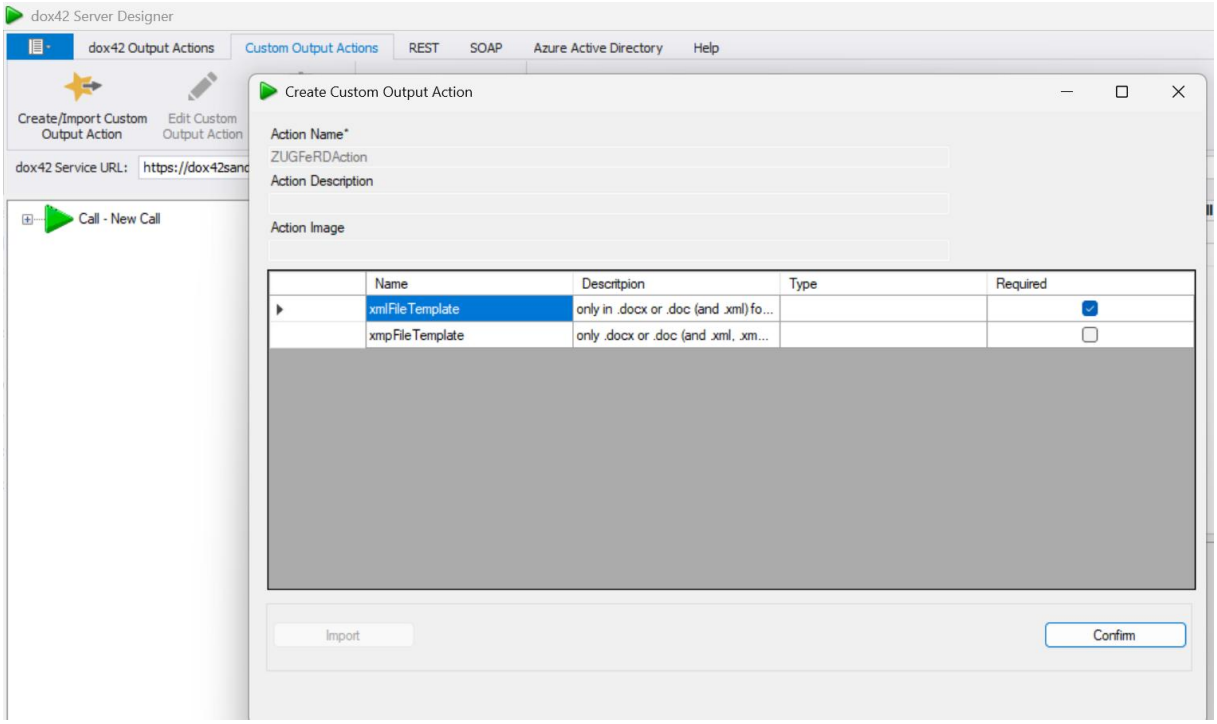
Since Version 5 you can generate e-invoices in ZUGFeRD format with dox42.

The easiest way to configure dox42 calls for ZUGFeRD invoices is with the dox42 Server Designer, which you can download from our. website

The ZUGFeRDAction is a CustomOutputAction and needs to be imported into the Server Designer beforehand. To do this, select the 'Create/Import Custom Output Action' button in the 'Custom Output Actions' tab of the dox42 Server Designer and upload this JSON file via the Import button.

Here you can also find the text of the JSON file:

{"actions":[{"id":"b9c258b8-417b-4480-a6d1-2530c278b0dc","name":"ZUGFeRDAction","icon":"","description":"","parameters":[{"id":"d98fbb3d-6e94-4bff-9aa7-b0e9dc0bc334","name":"xmlFileTemplate","description":"only in .docx or .doc (and .xml) format","type":"","required":true,"Value":""},{"id":"2c7c993c-2a81-45fb-b7f0-ffdf63a89393","name":"xmpFileTemplate","description":"only .docx or .doc (and .xml, .xmp) format","type":"","required":false,"Value":""}]}]}



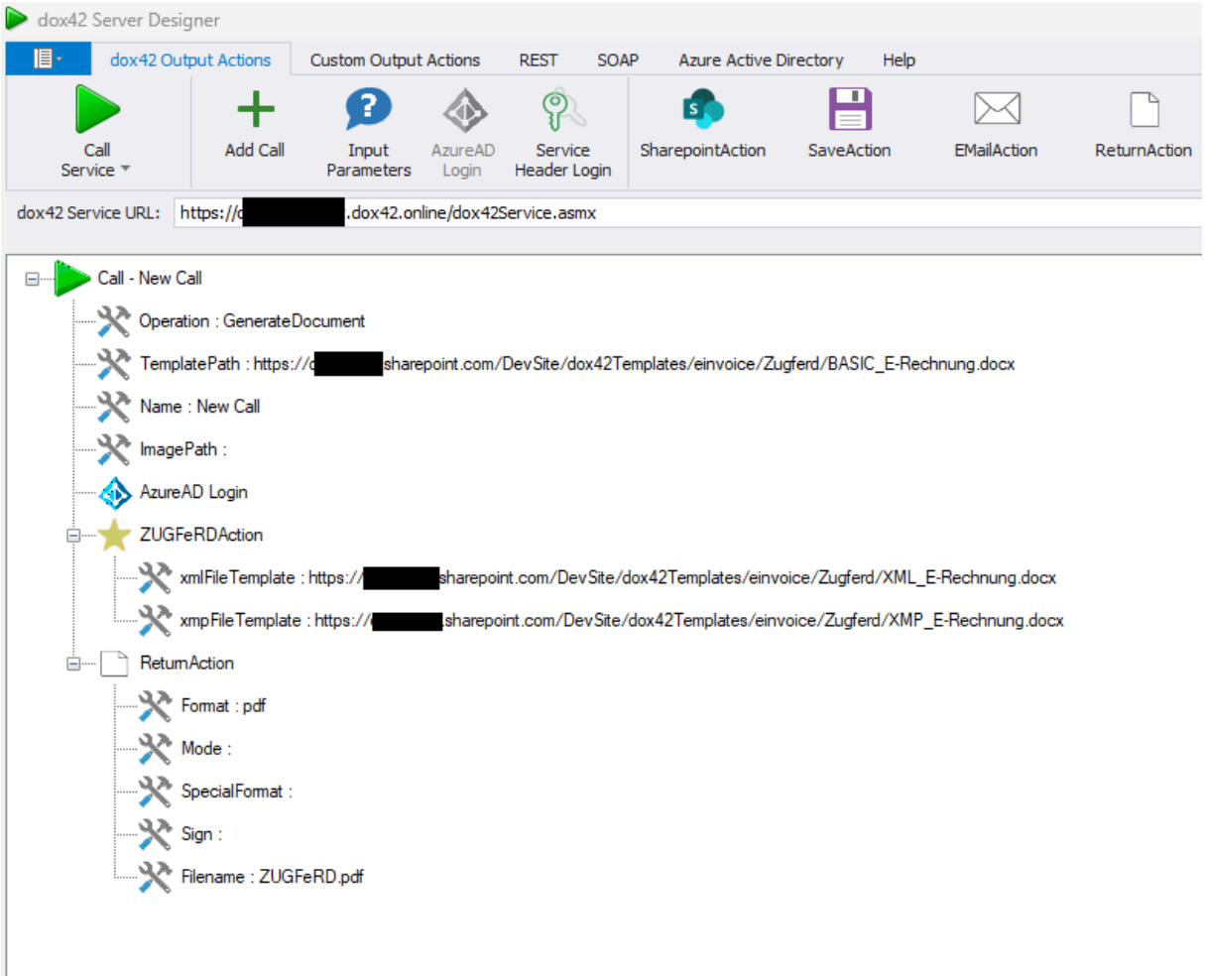
This process adds the 'ZUGFeRDAction' with the following fields:

Parameter name	Reequirement	Effect
xmlFileTemplate	mandatory	The path tot he hybrid XML template.
xmpFileTemplate	optional	The path to the XMP meta data.

In the next step, you can set up the dox42 service call for your ZUGFeRD file. Follow these steps:

Add the ZUGFeRDAction to your call with the paths to your XML and the optional XMP file. The ZUGFeRDAction is a custom modify action that needs to be called in conjunction with another dox42 output action. dox42 modify actions are used to enrich generated documents with additional data, such as an embedded XML and/or XMP file in this case. Therefore, add a dox42 ReturnAction or another output action to the call so that the result file is returned, saved, or, for example, sent.

You can validate the result of your ZUGFeRD invoices at https://www.portinvoice.com.
Here is an example of how you can configure your call in the Server Designer:


dox42 Server Designer
dox42 Output Actions
Custom Output Actions
REST
SOAP
Azure Active Directory
Help
Call Service
Add Call
Input Parameters
AzureAD Login
Service Header Login
SharepointAction
SaveAction
EMailAction
ReturnAction
dox42 Service URL: https://[redacted].dox42.online/dox42Service.asmx
Call - New Call
Operation : GenerateDocument
TemplatePath : https://[redacted]sharepoint.com/DevSite/dox42Templates/einvoice/Zugferd/BASIC_E-Rechnung.docx
Name : New Call
ImagePath :
AzureAD Login
ZUGFeRDAction
xmlFileTemplate : https://[redacted]sharepoint.com/DevSite/dox42Templates/einvoice/Zugferd/XML_E-Rechnung.docx
xmpFileTemplate : https://[redacted]sharepoint.com/DevSite/dox42Templates/einvoice/Zugferd/XMP_E-Rechnung.docx
ReturnAction
Format : pdf
Mode :
SpecialFormat :
Sign :
Filename : ZUGFeRD.pdf

Configuration parameters

The following parameters can be configured for the dox42 server in the AppSettings of the web.config.

- **LicenseFilePath**: Indicates the path to the location of the license file. In case no license file exists, the dox42 server will contact the dox42 license server in order to verify the license key. If successful, the server will write the license file to the location under the path indicated in the configuration.
- **LicenseKey**: license key of the dox42 server
- **MailServer**: email server for sending emails (optional)
- **MailServerPort:** email server port (optional)
- **dox42ServerMail**: sender's default address of the dox42 service (optional)
- **MailServerPassWord**: password for the dox42 email server (optional)
- **SendErrorsToMail**: standard email address for error messages (optional)
- **Certificate**: certificate, used for signing (optional)
- **CertificatePassWord**: certificate password (optional)
- **CheckServiceHeader (true/false):** Indicates, whether the service header is to be analysed or not. If "false", no service header needs to be included.
- **UserPermissionODBCConnectionString:** ODBC connection string to a data base, which contains user permissions
  (mandatory if CheckServiceHeader = true)
- **CheckUserPermissionSQL:** The select statement, which is executed on the UserPermissionODBCConnectionString data base. The statement has to return 1 or "true" in the case of a successfully authorised user. Within the statement, the %username% and %password% tags will be replaced with the values, included in the header.
  (mandatory if CheckServiceHeader = true)
- **ThrowExceptionOnError (true/false):**
  true: The service returns an exception on error.
  false (or not present): An error will be returned as ResultMessage.
  This configuration parameter is available since build 3.3.0.8.
- **CustomErrorMessagesFile:**
  Path of a XML Files containing Custom Error Messages (optional). Custom Error Messages allow for replacing Error Messages on the basis of a text search. Please find a sample later in the chapter.
- **HyphenationDictiorariesFile:**
  Path of a XML Files containing containing Hyphenation Dictionaries (optional). Please find a sample later in the chapter. (Since version 3.4.3.10)
- **CheckTrustedTemplateLocations (true/false):**
  true: The path/link of the template or data map must start with one of the paths/links listed in the section „trustedTemplateLocations". If not, the call is cancelled.
  false (or not present): No check is performed.
  CAUTION: if you switch off this check, everybody who is allowed to call your dox42 service can generate templates/datamaps from any location including unknown URLs on your dox42 server. We strongly recommend to make sure this check is always switched on for security reasons!

This configuration parameter is available since build 3.3.4.
List the trusted template locations in the section „trustedTemplateLocations“.

- **TempFolder:**
  Folder to be used by the dox42 Server to store temporary files. If not configured the Windows standard Temp Folder is used.
  This configuration parameter is available since version 3.5
- **RESTInterfaceReturnsFullErrorInfoTempFolder:**
  If true more detailed error information is returned for the dox42 REST call.
  This configuration parameter is available since version 3.6.1.
- **RESTInterfaceReturnsHTTPError500 (true/false):**
  if true the http Statuscode 500 will be set on error for the dox42 REST call.
  This configuration parameter is available since version 4.0.1
- **SOAPServiceReturnsFullErrorInfo (true/false):**
  If true more detailed error information is returned for the dox42 SOAP call.
  This configuration parameter is available since version 4.1.1.0
- **SaveActionEnabled (true/false):** (This parameter is available since build 4.3.2)
  true: you can use the dox42 SaveAction to store generated documents on your server
  false: you cannot use the dox42 Save Action and an error will be returned.
- **RedirectURLEnabled (true/false):** (This parameter is available sind build 4.3.2)
  true: You can add the parameter &RedirectURL=https://www.yourURL.com to your calls to redirect to a certain URL after document generation
  false: The RedirectURL parameter will throw an exception, you will not be redirected
- **MailMethod:** (This parameter is available sind build 4.3.3)
  ExchangeOnline: Needs to be used, if you send mails via Microsoft Exchange Online. Read the EMailAction section for more information.
  onasposeerrortrymailkit: uses Aspose to send mails, if Aspose method fails, MailKit is used to send dox42 emails
  aspose: uses Aspose to send mails
  onmailkiterrortryaspose: uses MailKit to send mails, if MailKit method fails, uses Aspose to send mails
  mailkit: uses MailKit to send mails
- **LoadLocalFilesFromTrustedTemplateLocationOnly (true/false):** (This parameter is available since build 4.4.1)
  If true, you can only integrate files from trusted template locations in your data sources and dynamic fields
- **AADLibrary (MSAL/ADAL):** (This parameter is available since build 4.6.1)
  Determines which Microsoft Authentication library is used. MSAL is the default value.

Example

```
<appSettings>

        <add key="LicenseFilePath" value="c:\dox42\dox42WebService\License" />
        <add key="LicenseKey" value="…"/>
        <add key="MailServer" value="smtp.1und1.de"/>
        <add key="dox42ServerMail" value="testserver@dox42.com"/>
        <add key="MailServerPassWord" value="…"/>
        <add key="SendErrorsToMail" value="monitoring@dox42.com"/>
        <add key="MailServerPort" value="587"/>
        <add key="MailMethod" value="onasposeerrortrymailkit"/>
        <add key="Certificate" value="c:\dox42\dox42WebService\Certs\Certificate.pfx"/>
        <add key="CertificatePassWord" value="…"/>
        <add key="CheckServiceHeader" value="true"/>
        <add key="UserPermissionODBCConnectionString"
              value="Driver={SQL Server Native Client 10.0};Server=…;Database=…;Uid=…;Pwd=…;"/>
        <add key="CheckUserPermissionSQL"
              value="SELECT COUNT(*) FROM Users where Username =
              '%username%' and Password = '%password%'" />
        <add key="ThrowExceptionOnError" value="false"/>
        <add key="CustomErrorMessagesFile" value="C:\dox42Server\CustomErrors.xml"/>
        <add key="CheckTrustedTemplateLocations " value="true"/>
        <add key="TempFolder " value="C:\dox42Server\WorkingDir"/>
        <add key="RESTInterfaceReturnsFullErrorInfo" value="false"/>
        <add key="RESTInterfaceReturnsHTTPError500" value="false" />
        <add key="SOAPServiceReturnsFullErrorInfo" value="false" />
        <add key="SaveActionEnabled" value="true" />
        <add key="RedirectURLEnabled" value="true" />
        <add key="LoadLocalFilesFromTrustedTemplateLocationOnly " value="false" />
</appSettings>
```

Custom data sources will be registered in a separate section under the name “customDataSources“.

```
<customDataSources>
        <add key="CSVDataEngine.CSVDataSourceParser"
              value="CSV Datenquelle;c:\dox42\dox42WebService\bin\CSVDataEngine.dll;"/>
</customDataSources>
```

Custom Output Actions Sources are registered in a specified section called “customOutputActions”.

```
<customOutputActions>
        <add key="MyCustomOutputActions.MyOutputAction"
        value=" MyOutputAction; c:\dox42\dox42WebService\bin\MyCustomOutputActions.dll;"/>
</customOutputActions>
```

Trusted template locations are registered in a specified section called “trustedTemplateLocations”.

```
<trustedTemplateLocations>
  <add key="local" value="C:\dox42Server\templates"/>
        <add key="sharepoint" value="http://www.mysharepoint.mycompany.com/dox42Templates"/>
 </trustedTemplateLocations>
```

Configure log settings in the file NLog.config. For a complete documentation of the logging framework NLog see www.nlog-project.org.

CORS Configuration

If a web-application calls the dox42 Server e.g. via Javascript XMLHttpRequest under a different URL than the application itself, you might get a Cross-Origin Resource Sharing (CORS) error.

Example:
Calling web-application: https://yourcompany.sharepoint.com/yoursite
dox42 Server: https://dox42server.yourcompany.com

To enable this scenario you could add a customHeaders section to your web.config, example:

```
<system.webServer>
    <httpProtocol>
      <customHeaders>
        <clear />
        <add name="Access-Control-Allow-Origin" value="*" />
        <add name="Access-Control-Allow-Headers"
              value="Content-Type, Accept, Authorization" />
        <add name="Access-Control-Allow-Methods" value="GET,POST,OPTIONS" />
        <add name="Access-Control-Expose-Headers"
              value= "Content-Type, Content-Disposition, Accept" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
```

See also:
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
http://www.w3.org/TR/cors/

Azure Active Directory Access

This is necessary to store dox42 Templates and Data-Maps on SharePoint Online.

ConfigSection:

```
<configSections>

       . . .

    <section name ="azureAD" type ="dox42.Core.Utils.AzureADSectionHandler,dox42Core"/>
</configSections>
```

Azure AD Section:

```
<azureAD>
    <add      resource365="https://mytenant.sharepoint.com"
              appID="…"
              tenant="…"
              clientKey="********************************" />
</azureAD>
```

Custom Error Messages Sample

```
<?xml version="1.0" encoding="utf-8"?>
<CustomErrorMessagesDS xmlns="http://dox42.com/CustomErrorMessagesDS.xsd">
  <CustomErrorMessage>
    <Test2SearchInExc>Error uploading file http://mysharepoint</Test2SearchInExc>
```

```
    <NewMessage>
SharePoint-farm not responding, please contact your administrator</NewMessage>
  </CustomErrorMessage>
</CustomErrorMessagesDS>
```

In case you edit this file using a text editor, please be sure to save in UTF-8 encoding.

Hyphenation Dictionaries Sample

```
<?xml version="1.0" standalone="yes"?>
<HyphenationDictionary xmlns="http://www.dox42.com/HyphenationDictionary.xsd">
  <Dictionary>
    <Culture>de-CH</Culture>
    <DictionaryPath>C:\dox42Server\hyph_de_CH.dic</DictionaryPath>
  </Dictionary>
</HyphenationDictionary>
```

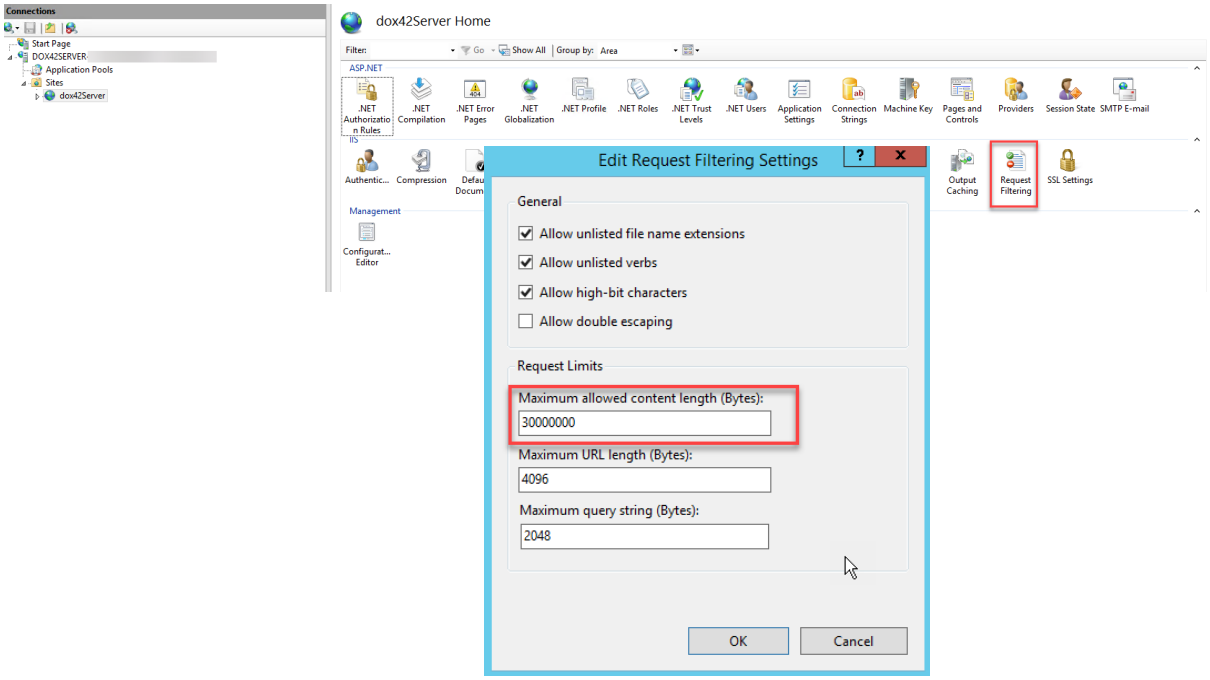
To use automatic hyphenation with the dox42 Word Add-In save such a file named HyphenationDictionary.config to C:\Users\<<USER>>\AppData\Local\dox42V1

Maximum size of request or time-out

Should you be sending large of requests to dox42, return large files or start a bulk letter, it might become necessary to increase request size or time-out. (The default ASP.NET time-out is 90 sec. After that, a longer call will be aborted by IIS.) Configure this in the web.config as well.

```
<system.web>
       <httpRuntime maxRequestLength="10000000" executionTimeout="1000"/>
</system.web>
```

When you work with the https method POST (base64) (applicable to dox42 D365 FO scenarios) and you are getting the error message – '413 Request entity too large", you can change the "Maximum allowed content length (bytes)" parameter on the IIS server. In ISS, navigate to Request Filtering, click on "Edit Feature Settings" and change the maximum allowed content length to 2147483647 from 30000000.



License Activation

Please enter the license key in the Web.config as „LicenseKey“.

On the first call the dox42 Server will contact the dox42 license server to activate your key and write a license file (*.lic) in the folder indicated under “LicenseFilePath”.

If the Server is only allowed to access the internet via Proxy, maybe the following entry in the web.config will help.

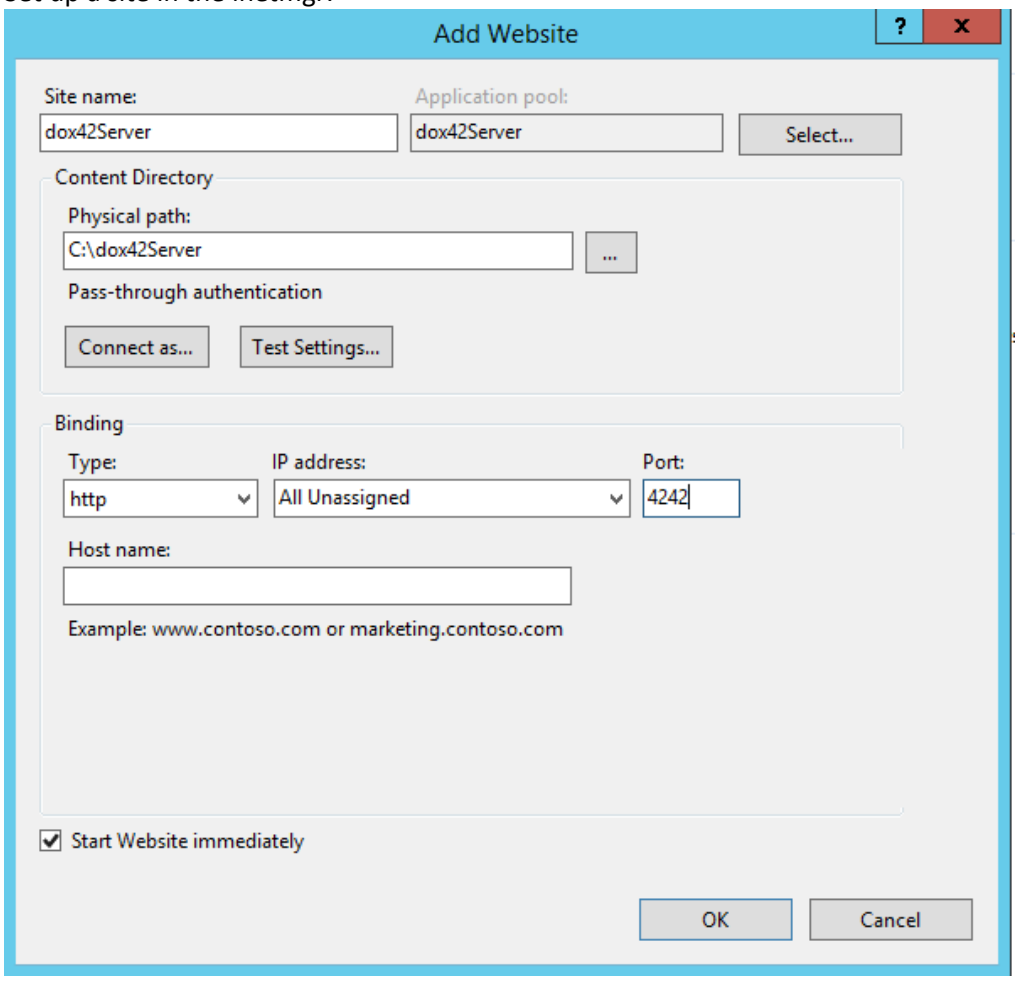
In case you cannot establish an internet connection from your server, please contact support@dox42.com, we will provide you with your license file.

Installation

The following section describes the installation on a Windows 2012 Server with IIS 7.5. You can of course use newer versions or a Windwos Azure Server to install dox42.

1. Copy the directory
   dox42ServerPack\dox42ServerVXXXX\dox42Server
   to
   C:\dox42Server
   (You can also choose a different virtual directory.)

2. Set up a site in the inetmgr:



   Please mind the port! You may use any other port, but bear in mind that the standard port (80) is often already in use by other applications e.g. Sharepoint. Of cource dox42 server supports https.

3. Copy your Aspose.Total.Lic file to the BIN Folder (C:\dox42Server\Bin) - not applicable for Trial Installations

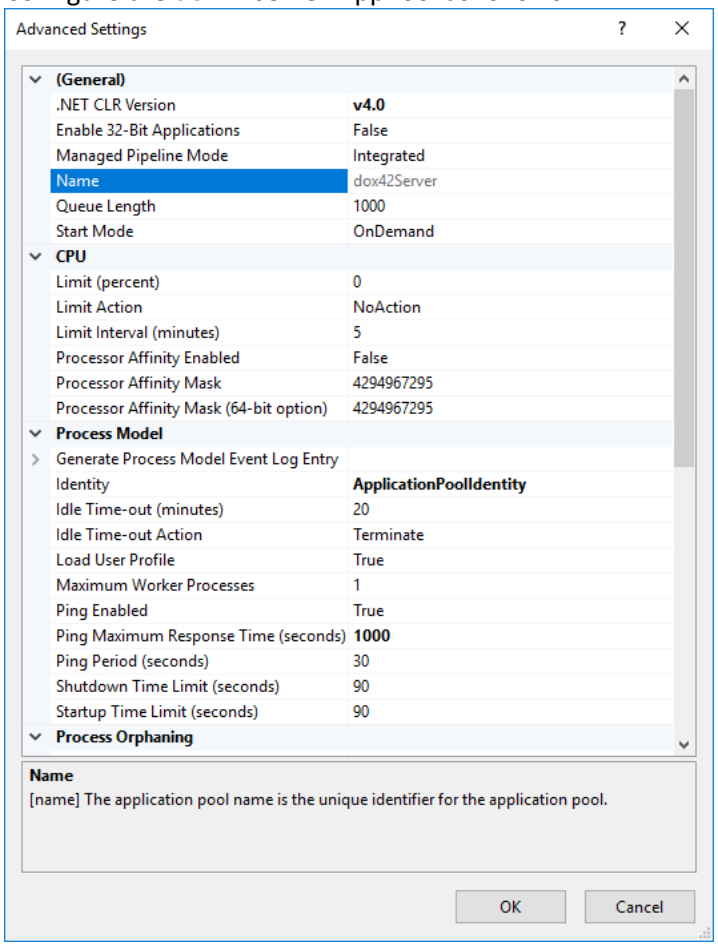
4. Install the NET 4.5.2 framework.

5. User Server Manager to activate ASP.NET (inclusive HTTP Activation)

6. If want to enable access to SharePoint Online (Office 365) for this dox42 Server, please install the Microsoft SharePoint Server 2013 Client Components SDK.
   Download: http://www.microsoft.com/en-us/download/details.aspx?id=35585

7. Run iisreset.

8. Configure the dox42 server-AppPool as follows:



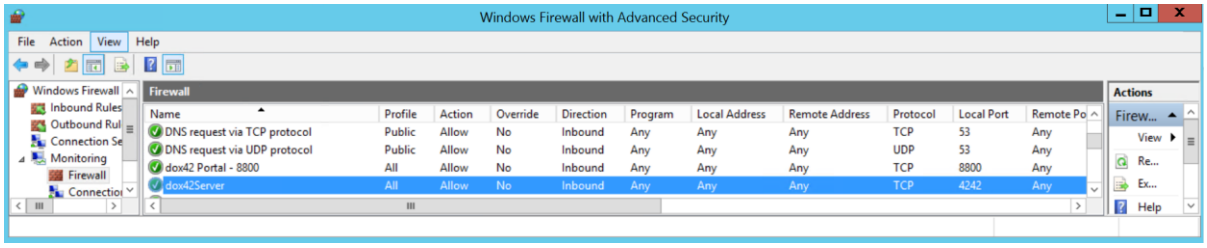
   Instead of ApplicationPoolIdentity you can also use a Service-User.

   In case you want to generate dox42 sample template which occasionally use MS Access (32-Bit needed) you might need to activate 32 bit applications.

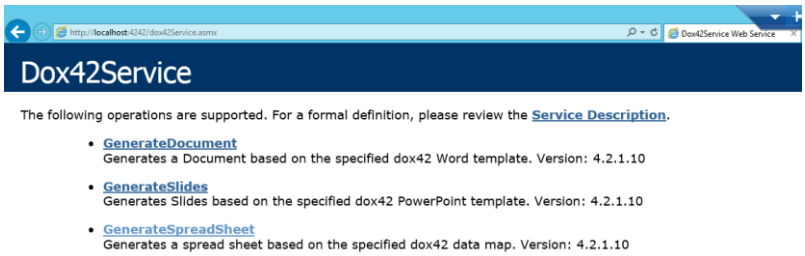
9. In inetmgr, change back to site > manage permissions
   Here, look for the user “IIS AppPool\dox42Server“ (Attention, use the local computer as search path!) or a dox42 service account and give this user read **and** write permissions to **C:\dox42Server**

10. Configure web.config (key, license-dir, mail server) and NLog.config (log-dir).

11. Do not forget to activate the port on your firewall:



12. Do a test to reach your dox42 web service via your Browser:
"http://localhost:4242/dox42Service.asmx"



13. Afterwards test generating a document with the dox42 Server Designer. Install the Server Designer application and open it. You can find it at C:\dox42ServerPackVXXX\dox42ServerDesigner_*.
Within the Server Designer, open the HelloWorld_ServerDesigner.config from C:\dox42Server\templates\HelloWorldTest and click on “Call Service” to test your dox42 server document generation.

    The dox42 Server Designer can be used for the easy configuration of your dox42 webservice calls. Please watch the #dox42class Video Tutorial for more information on how to use the Server Designer.

14. Be sure to use adequate Access Control to protect your dox42 Service against unauthorized use.

15. In order to use AzureAD you need an HTTPS certificate installed. In case you need help how to do this, watch this video: https://youtu.be/HHU46l6G2oc

Upgrade

To Upgrade a dox42 Server please perform the following steps:

1. Make a backup of the dox42 Server folder, so you can roll back in case anything goes wrong.
2. Delete all files in the bin folder EXCEPT Aspose.Total.lic and all dll-files, that belong to your custom data sources and output actions. Custom data sources and output actions are registered in the web.config, e.g. dox42SAP, dox42AX, dox42DynamicCRM, etc. Please see the documentation of each module to get a list of the necessary libraries.
3. Copy all files from the bin folder of the dox42 Server Pack to the bin folder on your server.
4. Copy all files EXCEPT dox42LIVE.config from the dox42LIVE folder of the dox42 Server Pack to the dox42LIVE folder on your server.

5. Check if Dox42Service.asmx and dox42RestService.ashx are present in your dox42 Server folder.
6. Check your web.config for new configuration options.
7. Restart your dox42 Server in IIS.
8. Test with dox42 ServerDesigner
9. That's all, you are done, go for a beer ☺

Support

Should you have any questions, please do not hesitate to contact support@dox42.com

Good luck with your dox42 server!