

DocuFlow

Version 80000

Installation
And
User's Guide

© 2022-2022 Mpowered Ventures Ltd.
301 – 1555 Fir St
White Rock, BC V4B 4B6
Canada
<http://www.mpowered.biz>

1. Installing DocuFlow

System Requirements

Web App (External - if in DMZ scenario - explained later in doc):

- .NET: Requires an IIS server, running .NET Framework v4.7.2 (note that v4.5 will work, but will not be as secure as v4.7.2 - mainly because 4.7.2 uses TLS 1.2 by default)

Web Services (Internal):

- .NET: Requires an IIS server, running .NET Framework v4.7.2 (note that v4.5 will work, but will not be as secure as v4.7.2 - mainly because 4.7.2 uses TLS 1.2 by default)

Tempest Licences:

- Prospero
- Land

Because these next 2 items may take some time, these are shown here right away so the appropriate technical specialist can get them set up, hopefully by the time everything else in this document is ready to go.

Technical specialist: SQL Server: enable Ole Automation

The TEST and LIVE databases will need to have Ole Automation enabled, if not already enabled. See the URL:

<https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/ole-automation-procedures-server-configuration-option?view=sql-server-ver15>

This is required because the process initiates a web service call to the DocuFlow web services from a T-SQL function triggered by a Prospero task status function.

Technical specialist: allowList DocuSign IPs

DocuSign Connect messages are sent from a variety of IP addresses, and your IT department will need to allowlist those IPs. See the URL:

<https://www.docusign.com/trust/security/esignature>

This is required because DocuSign sends “Connect” messages back to the DocuFlow web service as each document signing process goes through its various stages.

Download the Install package

Go to www.mpowered.biz and click on Downloads. Here you will find links to various setup ZIPs that match recent versions of Tempest. Click on the Download link for the most recent version. This will download the ZIP package, which you can then extract into a working directory on your web servers (external and internal will need to be installed at the same version).

Contents of the ZIP package

Once the ZIP package is extracted to a working directory, you will find this structure:

```
\Docs  
\Dotnet  
\Integrations\DocuSign
```

Create database user MpoweredWeb

Create a user named MpoweredWeb in each Tempest database (usually LIVE and TEST) that you wish to access with DocuFlow. (You may already have created MpoweredWeb during the install of other Mpowered products.)

Grant database user MpoweredWeb database access permissions

Grant the table permissions found in \Docs\dbgrants.txt to the database user MpoweredWeb using your database management console. The table permissions need to run in each database (for example, Test and Live) that you will run DocuFlow against. It's important to run the grants provided in the dbgrants.txt for each new version of DocuFlow, as they can change – as well as after every Tempest patch/update.

Install the .NET web app

The \Dotnet directory contains the .NET web services required for DocuFlow. Your installation will be one of the following two scenarios:

DMZ scenario

If your machine infrastructure involves a DMZ, separate from your internal protected zone, you will need a server to host the web services on the DMZ as well as a server to host the web services in the internal protected zone. In this scenario, let's call it the DMZ scenario, the DMZ server will forward requests from the Internet (e.g. DocuSign Connect messages) to the internal web server. The external web server and internal web server must be running Mpowered DocuFlow web services at the same version.

WAF scenario

If your machine infrastructure involves a "web application firewall", you will most likely need to install the web services on your internal protected zone only. You will only need to host the web services on a single server in the internal protected zone. In this scenario, let's call it the WAF scenario, Internet requests (e.g. DocuSign Connect messages) will be routed directly to the internal web server.

Internal Web Server (needed for both DMZ and WAF scenarios)

On your internal web server, create a home directory for the Mpowered .NET web services if you don't already have one... something like:

```
C:\inetpub\wwwroot\Mpowered\DocuFlow-80000
```

or:

```
C:\inetpub\wwwroot\Mpowered\DocuFlow-80000TEST
```

for a Test version. Copy the entire \Dotnet\WebServices\Redmond directory from the download here. Now on your internal web server, you should have this structure:

```
...\wwwroot\Mpowered\DocuFlow-80000\  
    bin\  
        DP80000WS.dll  
    DocuFlow.asmx  
    Web.config.internal.txt
```

Now edit the Web.config.internal.txt file and look for a section with the tag <connectionStrings> near the bottom. Here you will see a sample connection string for SQL Server named "MpoweredSQL".

With the connection string you will use, edit it so that YOURHOST becomes the server name where the Tempest database lives, and INSTANCE becomes the name of the database instance. Also, change the Password= to the MpoweredWeb password you created earlier. (NOTE: the password is entered in clear text here – this file should be secured so that only people with proper permissions can view this file. Contact Mpowered for more info about encrypting the config file if you wish additional security.) If you don't know the server name or password values, you may have to talk with your Database Administrator.


Note: you can have multiple connection strings in this file, for example you could have an MpoweredSQLProd and an MpoweredSQLTest connection string each pointing to the Production and Test Tempest databases, although the preferred method is to have Live and Test web services versions in separate directories. In the DMZ scenario, when you set up the external web server below, you will choose which DSN (connection string) to use.

Very important!

Save and exit. Rename the **Web.config.internal.txt** file to **Web.config** (make sure you are viewing file extensions, because it won't work if the file still has a .txt extension!)

If you have setup up other Mpowered apps, or are upgrading from a previous DocuPro-80000 version, MpoweredApps will already be there, so you can skip this step.

Now fire up IIS Manager on the internal web server. Browse into Application Pools, and right-click and choose Add Application Pool. Create a new pool named "MpoweredApps" using .NET CLR Version v4.0.30319 (if you do not have this version, you will need to install MS .NET Framework 4.7.2 on this machine), Integrated, Start application pool immediately ON. Click on the newly created pool, and browse to Advanced Settings on the right side menu. Make sure that Enable 32-Bit Applications is set to True, and click OK.

Now on the left tree, browse down to Sites > Default Web Site > Mpowered and right-click on DocuFlow-80000. Choose Convert to Application. Keep the Alias as DocuFlow-80000, but select Application pool MpoweredApps, and click OK. This should change the icon in the tree to: .

Now click on DocuFlow-80000 and click the Content View button on the bottom of the IIS window, and then right-click on DocuFlow.aspx, and choose Manage Application > Browse. The default browser should appear with the DocuFlow .NET services listing **, containing links for AA_ServiceInfo, AC_DatabaseTest, etc. Click on AC_DatabaseTest, and just hit Invoke. You should get an XML page that says "SUCCESS: Found nnnn rows in the cd_tasks table". This means that the DSN was set up correctly, and we are getting a connection to the Tempest database.

If you get the message “Timeout expired. The timeout period elapsed prior to completion of the operation or the server is not responding.” you may be able to solve the issue by running “exec sp_updatestats” on the database.

That completes the set-up of the internal web server.

External Web Server (needed for the DMZ scenario only)

On your external (on the DMZ) web server, create a home directory for the Mpowered .NET web app... something like:

```
C:\inetpub\wwwroot\Mpowered\DocuFlow-80000
```

or:

```
C:\inetpub\wwwroot\Mpowered\DocuFlow-80000TEST
```

for a Test version. Copy the entire \Dotnet\Client directory from the download here. Now on your external web server, you should have this structure:

```
...\wwwroot\Mpowered\DocuFlow-80000\  
    bin\  
        DP80000.dll  
    favicon.ico  
    GetBlade.aspx  
    ...etc  
    Web.config.external.txt
```

If you are upgrading from a previous version, just copy the Web.config from there.


Very important!

Now edit the Web.config.external.txt file and look for a section with the tag <appSettings> near the bottom. Here you will see a “webservice” key. It is the value that you must edit to point to the web services location on the internal web server (through the firewall). You may need to get your firewall expert to help you figure this one out. In most cases, you will simply need to change {ip} to the ip address of the internal web server (as seen from the DMZ).

Save and exit. Rename the **Web.config.external.txt** file to **Web.config** (make sure you are viewing file extensions, because it won’t work if the file still has a .txt extension!)

If you have setup up other Mpowered apps, or are upgrading from a previous DocuPro-80000 version, MpoweredApps will already be there, so you can skip this step.

Now we need to fire up IIS Manager on the external web server. Browse into Application Pools, and right-click and choose Add Application Pool. Create a new pool named "MpoweredApps" using .NET CLR Version v4.0.30319 (if you do not have this version, you will need to install MS .NET Framework 4.7.2 on this machine), Integrated, Start application pool immediately ON. Click on the newly created pool, and browse to Advanced Settings on the right side menu. Make sure that Enable 32-Bit Applications is set to True, and click OK.

Now on the left tree, browse down to Sites > Default Web Site > Mpowered and right-click on DocuFlow-80000. Choose Convert to Application. Keep the Alias as DocuFlow-80000, but select Application pool MpoweredApps, and click OK. This should change the icon in the tree to:  (you may have to refresh to see the icon).

A generic alias is highly recommended to save time in the future when new releases come out.

On the Internal server (in the WAF scenario), or the External server (in the DMZ scenario), we are going to additionally create a "generic" alias that will point to this version, and can point to new versions (as they are released in the future) so that any references to URLs will not need to be changed in order to run the most-recent version of the application. This is handy when creating the DocuSign Connect call-back later as we will not need to change it as updates to DocuFlow are made.

In IIS, right-click on the Mpowered node, and choose "Add Application...". Name the Alias "DocuFlow" (or "DocuFlowTest" for a Test version), set the Application pool to "MpoweredApps", and under Physical path use the [...] button to browse to the ...\\wwwroot\Mpowered\DocuFlow-80000 directory used above. You should now see a node like this:

 DocuFlow

under the Mpowered node (you may have to refresh to see it).

Releasing new versions.

As you upgrade in the future, and after you have tested the new version using the DocuFlow-nnnnn application, you can edit this DocuFlow alias to point to the new version's physical path (click on the DocuFlow alias > Advanced Settings > Physical Path). Edit the alias once you are ready to "release" the new version and existing URLs such as the DocuSign Connect call-back will seamlessly be transitioned to the new version..

The Web.config files

The Web.config file on the internal server contains the configuration settings for managing the Tempest impersonated userId as well as mail and DocuSign settings.

The bulk of the changes needed will be in the **internal** Web.config. This file is documented with comments, and can guide you if you are going to DIY the install. Mpowered is always available for consultation, as this file can be tricky to understand and get right. The Web.config lets you configure a production and test setting for DocuSign. This is helpful, because you should always set up and test any changes to the Web.config in the TEST system. Once you are ready to set up LIVE, you can simply copy the Web.config file, and change the connectionString setting to point to the LIVE system. Once you have set up the LIVE Web.config, it is a good idea to copy it back to the TEST system, and change the TEST system's connectionString back to point to the TEST database.

The most important advice here is to ALWAYS have a Test environment for DocuFlow, and test any changes to the Web.config in Test first. Because DocuFlow operates behind the scenes, there are no error messages or other information shown, and so if you are running into issues - the debugglobal setting can be set to Y, and a z-debug.txt file will be created in the root web service directory as each web service is called. You will need to give 'Users' write permission into the directory. Another piece of standard advice: make a backup copy of the working Web.config file before you start making changes!

Once you are 100% sure that the Test site is working as expected, then promote the Web.config to Live. Note that **as soon as** you put a new Web.config in place, IIS will begin to use it immediately, and if there are issues or errors in the file, users will get those immediately as well – which obviously is undesirable.

AE_EmailTest operations

There is a helpful Web.config checker for your internal web services AE_EmailTest. You can access these operations while on the **Internal** web server's IIS Manager, right-clicking on DocuFlow-80000, and choosing Manage Application > Browse.

DocuFlow

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [AA_ServiceInfo](#)
Returns basic service info (no forwarding), indicating that the web service is registered on the machine correctly. F must have write permission on the the web service directory).
- [AB_ServiceTest](#)
Test basic service operation, including forwarding (if turned on). Leave all values empty for default test. If you are Internal Server Error.' then 'Users' must have write permission on the the web service directory on the forwarder.
- [AC_DatabaseTest](#)
Test database connection (defaults to SQL Server using DSN 'MpoweredSQL'). To test Oracle, enter a string like '<| 'MpoweredORA' with your Oracle DSN. Leave all values empty for default test.
- [AE_EmailTest](#)
Checks appSetting 'app-000.090' to test if emails are going through. This method can only be invoked on the Inter
- [AF_JSONTest](#)
Check this - in the classic External/Internal configuration - from the External server to see whether pure JSON con to the Internal Server. If this method fails with a '(500) Internal Server' error, it may be that the Internal Server i
- [AZ_ReleaseNotes](#)
Displays release notes for each version/package.
- [fr_docusign](#)
- [to_docusign](#)

AE_EmailTest will check that an email can be sent to the email address defined in app setting app-000.090. It is really important to make sure that emails are going through on the Live system! The most common issue with not receiving emails is that the Internal server does not have a firewall opening to the server/port defined in the Web.config.

The DOCUFLOW user in Tempest

The Internal Web.config setting app-000.020 (usually DOCUFLOW) defines the Tempest user the DocuFlow app will user-stamp records for all of its Tempest database operations as well as for assigning Tempest security, and it requires a certain amount of setup to ensure it has access. 1. Create a Tempest Resource with the name matching setting app-000.020 (usually DOCUFLOW). 2. The DOCUFLOW user does not need to be Database Authenticated or anything else on line 2.

Tools/Resource/Docuflow New

Record # 1 User Id DOCUFLOW

Name DOCUFLOW

Main Workgroups Contact Details User Tags Notes Attachments Resource Changes Log Roles Security Permission List

Resource Number 164 Last Login

Database Authenticated NO Network Authenticated NO Webservice Access NO Webservice Direct Access NO

Alias DOCUFLOW -Login Enabled Disabled

Resource Type INTERNAL

Organization Start Date Jan 1, 2022 Stop Date

Free Form Location

On the Workgroups tab, add the Workgroup(s) that the DOCUFLOW user will be a member of. This will depend on the task type(s) that you choose (or create) to handle the to/from DocuSign process, so you may not know this yet, but you will need to come back to this once the task type is chosen or created (discussed in a following section).

Tools/Resource/Docuflow New

Record # 1 User Id DOCUFLOW

Name DOCUFLOW

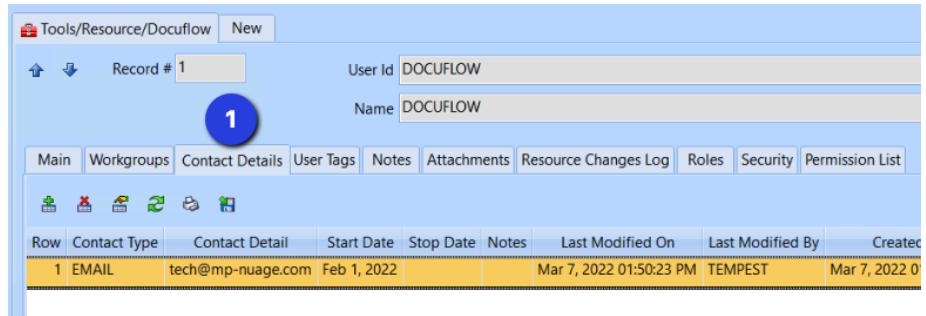
Main Workgroups Contact Details User Tags Notes Attachments Resource Changes Log Roles Security

1

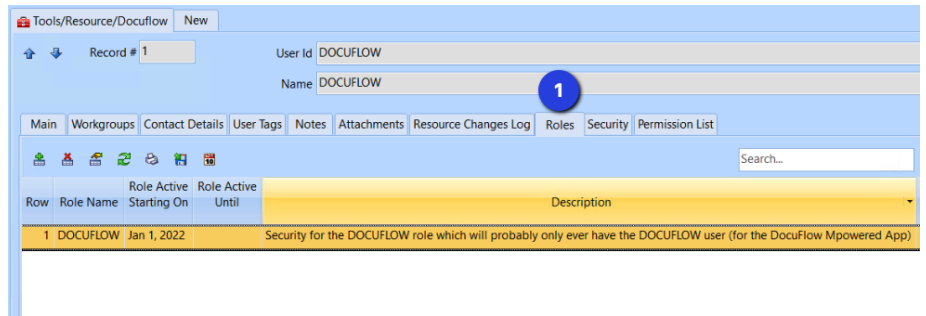
Row	Workgroup	Workgroup/Resource Start Date	Workgroup/Resource Stop Date	Workgroup/Resource Last Modified On	Workgroup/Resource Last Modified By
1	APP CENTRE	Jan 1, 2022		Feb 16, 2022 01:00 PM	TEMPEST

2

The Contact Details tab should have at least an EMAIL set up for the DOCUFLOW user. This will usually be a technical resource in your organization:



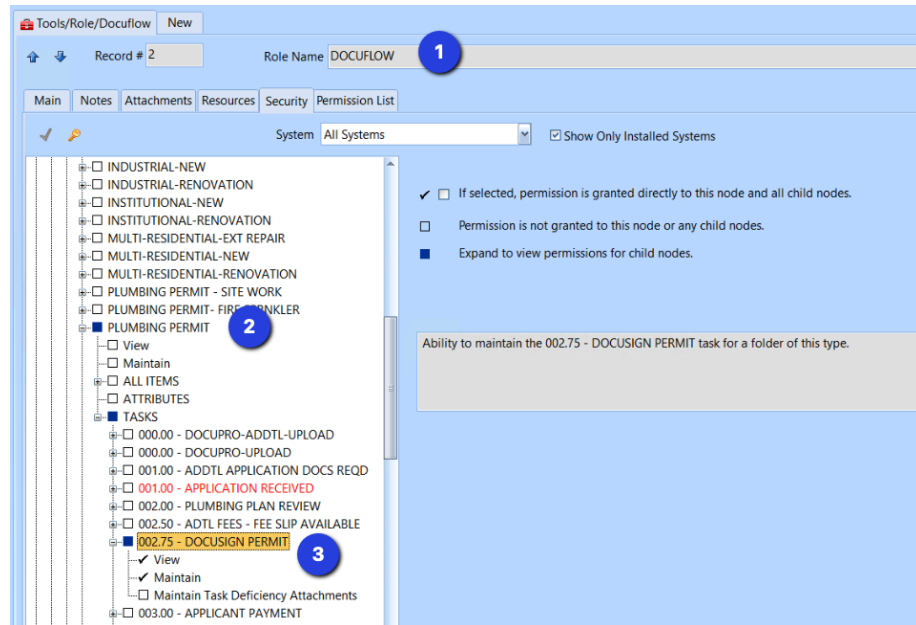
Optionally, the Roles tab will contain the Role for the DOCUFLOW user, in this case a newly-created DOCUFLOW role to hold the DOCUFLOW user:



Security Nodes for DocuFlow

As mentioned previously, you can optionally create a Role for the DOCUFLOW user to reside in, or you can give the security directly to the DOCUFLOW resource.

In the example below, the DOCUFLOW role (containing only the DOCUFLOW resource) has been granted View and Maintain security for the PLUMBING PERMIT folder type, on the task DOCUSIGN PERMIT.



You will need to assign this security for any folders/tasks that are set up to do the DocuSign process with DocuFlow, otherwise the call-back messages (“Connect” messages) will fail because DocuFlow requires View/Maintain security on any task it will be updating with the associated call-back actions.

When setting up new folders/tasks for the DocuSign process, it is important to remember to add this security, otherwise tasks will not have their statuses automatically updated by DocuFlow. As soon as DocuFlow sees this situation, an email will be sent to the technical contact defined in the Internal Web.config setting app-000.90 so that it can be corrected in Prospero configuration to allow future Connect messages to proceed for this task type. Unfortunately, if a Connect message is “lost” in this way (due to insufficient security), it cannot be re-sent, and the task status may need to be set manually. The security should be added as soon as possible, so that the “Completed” status can download the signed document from DocuSign and add it to the Prospero folder attachments.

Folder task configuration

Below we have set up a task named DOCUSIGN PERMIT on our PLUMBING PERMIT folder type:

It's a good idea to create a new task for the DocuSign process, because you usually want to use Statuses and Functions dedicated to the DocuSign process. Here are the 6 statuses we have added:

Row	Status	Last Modified On	Last Modified By
1	COMPLETED	Jan 25, 2022 02:46:41 PM	TEMPEST
2	FAILED	Feb 23, 2022 06:17:52 PM	TEMPEST
3	IN-PROGRESS	Jan 31, 2022 05:45:42 PM	TEMPEST
4	NA	Jan 25, 2022 02:46:41 PM	TEMPEST
5	PENDING	Feb 3, 2022 11:54:30 AM	TEMPEST
6	REJECTED	Mar 22, 2022 05:20:15 PM	TEMPEST

The NA status is a long-time standard Tempest recommendation to have a task status that allows us to indicate that we intentionally did not want to use the DocuSign process for a particular folder.

Before explaining what the other 5 statuses (COMPLETED, FAILED, IN-PROGRESS, PENDING and REJECTED) will be used for, let's take a step back and look at the big picture of the document signing interaction between Tempest, DocuFlow and DocuSign in this process.

DocuSign uses a container, for all documents to be signed, called an "Envelope". In our example, we will be sending an Envelope to DocuSign containing one Tempest-generated permit to be signed. The permit sent for signature is always the most-recent permit generated using the Details button on the Folder. Once generated, the permit is stored under the Archives node of the folder. Again, there can be multiple versions here as the permit details can be changed, but DocuFlow will always only use the most-recently generated permit.

At the start of the process, a Tempest user will set a folder's DOCUSIGN PERMIT task to a chosen status, e.g. IN-PROGRESS which triggers the action to send various pieces of information about the folder and task to DocuFlow. DocuFlow takes that information and creates a new Envelope in the format needed by DocuSign, bundles in the archived permit and sends the Envelope to DocuSign to securely manage the signing process. DocuSign sends the Envelope to the defined email recipient (typically the APPLICANT contact email address) and sets its internal Envelope status to "Sent". As soon as DocuSign has sent the Envelope to the recipient for signing, DocuSign will send a "Connect" message to DocuFlow with the "Sent" status. DocuFlow will then update the Tempest task status and perform any status-related functions. In our example, the "Sent" status updates the task status from IN-PROGRESS to PENDING.

So now the recipient gets an email from DocuSign on behalf of the sender (you, the City) inviting them to review and sign the document. The recipient can either: sign the document which changes the Envelope status to "Completed" and downloads the signed document as a folder attachment; or decline signing the document (maybe they thought something was wrong on the document and want you to modify it) which changes the Envelope status to "Declined"; or not ever sign the document (maybe they forgot about it?) which changes the Envelope status to "Expired". In any case, DocuSign will send back the document's new DocuSign status to DocuFlow when it happens via a Connect message. For these statuses, "Completed" makes sense to set the task status to COMPLETED, "Declined" to REJECTED, and "Expired" to FAILED. We can set up the status mapping in any way we want, but Mpowered recommends this DocuSign-to-task-status mapping.

Because the Tempest task status is kept up-to-date by DocuSign Connect messages, IntelliSearch queries can be used to determine

which folders can move on in the permit process (COMPLETED status), or determine which folders may need some additional followup (REJECTED and FAILED statuses.)

Now, let's take a look at some example functions tied to the DOCUSIGN PERMIT task statuses:

Row	Status	Function	Function Item
1	COMPLETED	SQL Function	ISSUED DATE (SET OR RESET) 3
2	IN-PROGRESS 1	SQL Function	DOCUSIGN PERMIT
3	IN-PROGRESS 1	Validation	DOCUSIGN CHECKS (ERROR)
4	PENDING	Send Email using Template	TASK STATUS 2

1. At the start of the process, when the DOCUSIGN PERMIT task on this folder type is set to IN-PROGRESS, Prospero does a Validation to ensure the permit is in the right state to begin the DocuSign process (i.e. has at least one archived permit, and an APPLICANT name and email), and then runs a SQL Function named DOCUSIGN PERMIT - which starts the Envelope creation/sending process as described above.

[The DOCUSIGN PERMIT and DOCUSIGN CHECKS (ERROR) Function Items are In-House, and Mpowered will work with you to create yours on your Tempest system once you have purchased DocuFlow.]

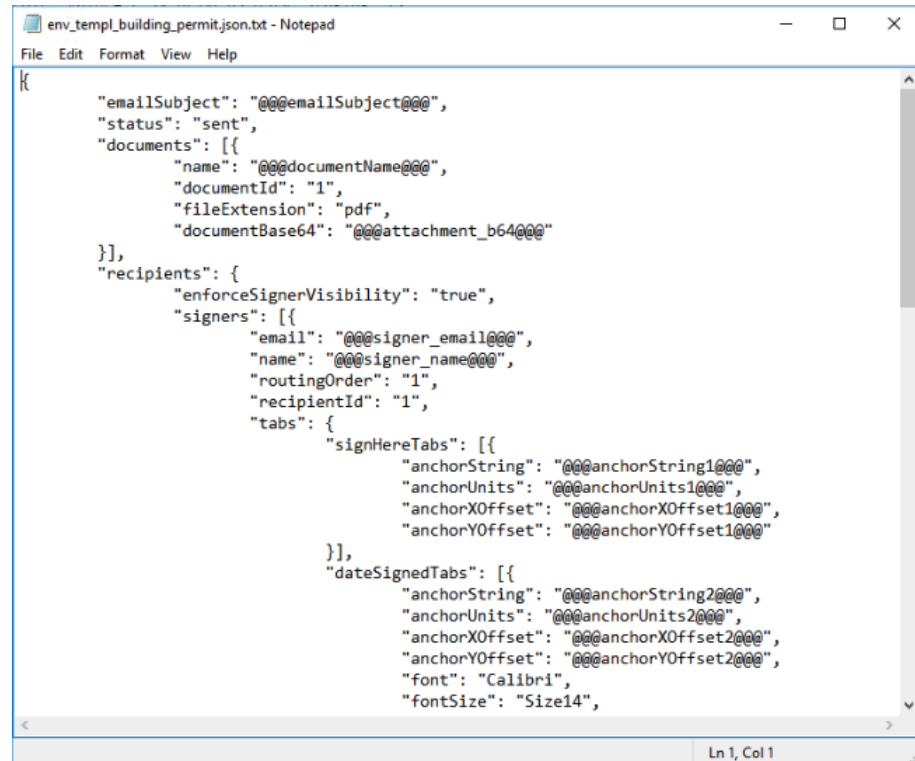
2. When DocuSign sends back the Connect message with the “sent” status, DocuFlow will use the status mapping to see the “Sent” DocuSign status and update the task status to PENDING. When that happens, we may want to send out an email as is shown with the sample TASK STATUS email template. Emails can be triggered by any task status update.

3. When DocuFlow gets the “Completed” status Connect message from DocuSign, DocuFlow sets the task status to COMPLETED, and sets the folder Issued Date, based on the SQL Function ISSUED DATE (SET OR RESET) Function Item we configured.

The envelope template file

When you first implement DocuFlow with Mpowered, you will receive an in-house DOCUSIGN PERMIT task function. Mpowered will work with you to customize this function. One of the things required in the task function is the location of an envelope template file.

The install directory \Integrations\DocuSign contains a sample envelope template (env_temp_l_building_permit.json.txt) that can be used as a starting point. Here is a part of that file:



```
env_temp_l_building_permit.json.txt - Notepad
File Edit Format View Help

{
  "emailSubject": "@@@",
  "status": "sent",
  "documents": [
    {
      "name": "@@@",
      "documentId": "1",
      "fileExtension": "pdf",
      "documentBase64": "@@@"
    }
  ],
  "recipients": {
    "enforceSignerVisibility": "true",
    "signers": [
      {
        "email": "@@@",
        "name": "@@@",
        "routingOrder": "1",
        "recipientId": "1",
        "tabs": {
          "signHereTabs": [
            {
              "anchorString": "@@@",
              "anchorUnits": "@@@",
              "anchorXOffset": "@@@",
              "anchorYOffset": "@@@"
            }
          ],
          "dateSignedTabs": [
            {
              "anchorString": "@@@",
              "anchorUnits": "@@@",
              "anchorXOffset": "@@@",
              "anchorYOffset": "@@@",
              "font": "Calibri",
              "fontSize": "Size14",
            }
          ]
        }
      }
    ]
  }
}
```

The envelope template has various places for replacements, for example @@@emailSubject@@@ which DocuFlow will replace with the actual email subject desired. The file is in JSON format, and based on specifications found on the DocuSign website, but if you are not comfortable with making changes in the file, it is probably best to consult Mpowered to see if the change you want to make is possible, and whether it will do what you intend.

A good strategy is to copy the supplied sample, and use it to start with, and if it looks like you want to make changes, contact Mpowered. Your envelope template file should be stored outside of any folder structure that IIS serves for best security, for example c:\DocuFlowAssets

For your TEST system, start by creating a DocuSign developer account to test the process

The steps needed to create a DocuSign developer account are beyond the scope of this document, but they are well-documented on DocuSign's website and elsewhere. It is recommended to create a generic "City" account for this purpose, not an account in a specific user's name. Once you have your developer account created (don't create a Trial account!, create a developer account), there are a few more steps to set up your account:

1. On DocuSign, in your developer account, Apps and Keys:
2. Create a new application named DocuFlow
3. Service Integration: Generate RSA, and copy the secret key generated to a file on your Internal web server (Mpowered will assist you to get this right). (Technical note: the RSA key is used by DocuFlow to generate a JWT, which is sent to DocuSign as part of the Envelope, and ensures that requests sent by DocuFlow on your behalf are authorized and secure. The private key file containing the RSA secret key should be stored outside of any folder structure that IIS serves for best security, for example c:\DocuFlowAssets)
4. Redirect URLs: set to <http://localhost>
5. Authorize all Tempest users to impersonate the DocuFlow app on DocuSign by browsing to the Authorization URL below (substituting your app's Integration Key from your newly-created application for 55b160e9-xxxx-xxxx-xxxx-f2f5412725dd). You will need the password for the account created above in order to authorize the impersonations. You will get a 404 error, but this is normal.

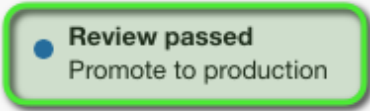
Authorization URL:

```
https://account-d.docusign.com/oauth/auth?response_type=code&scope=signature
impersonation&client_id=55b160e9-xxxx-xxxx-xxxx-f2f5412725dd&re
direct_uri=http://localhost
```

Mpowered will assist you every step of the way with all of the above steps.

Before going Live

DocuSign ensures that requests and callbacks for any application defined in an account pass at least 20 successful runs. Once you have completed this, the DocuSign application (“DocuFlow” usually) can be set to a status that requests Go-Live verification by DocuSign. DocuSign verifies that the requests and callbacks meet their requirements, and sets the application to Review passed as shown on this development application details screen:

Environment	Go Live Status
Development	 Review passed Promote to production

Of course, you will need a production account on DocuSign, and this process is something that you will negotiate with DocuSign. When your Live account is ready, please contact Mpowered to get the Live system configured and operational.