Setup and Configurations

- Connecting to UtopiaNuget
- Steps to setup pg_partman
- How to run and Release Mobile
- SonarCloud and SonarLint Setup Steps
- How to Run WOPI Validator Tests on O365
- AWS DMS Setup
- Previewer Setup for Local Environment
- OCR Worker Local Setup
- Windows Services
- Setup/Configurations for Mac Mini
- Office Addins Terminal Services Environment
- Utopia Local DB Refresher
- Generating a DB Connection String
- How to Sign and Notarize a New Build for the Mac
- How to Renew a Certificate for Mac
- Local Testing

Connecting to UtopiaNuget

Link to UtopiaNuget:

https://dev.azure.com/eFileCabinet/Utopia/_packaging?_a=feed&feed=UtopiaNuget (alternatively try this:

https://pkgs.dev.azure.com/eFileCabinet/Utopia/_packaging/UtopiaNuget/nuget/v3/index.json)

First connect to feed by pressing the Connect to Feed button at the top. Click on the Visual Studio option. Follow steps.

(Once you've created the Nuget Package Source, try to build the project and see if it succeeds before moving trying the below stuff)

Once connected to feed then go ahead an find the nuget that is needed and click on it. Then copy the nuget command and go into visual studio and go to Tools-Nuget Package Manager-Package Manager Console. Then paste the nuget command into the console and run.

If issue of 401 comes up use link to resolve: https://blog.rsuter.com/how-to-use-a-private-vsts-nuget-package-feed-with-the-net-core-cli/#:~:text=If you have a.NET Core project which references,status code does not indicate success%3A 401 (Unauthorized) .

UtopiaNuget on Mac

If you're doing this on mac (sorry), then the process is slightly different. See this other wiki page.

Steps to setup pg_partman

Steps to setup partman

** Make sure you don't have the postgres service running on windows **

Use WSL Ubuntu :

- It may need some additional setup (enable virtualization, etc)

Install postgresql on ubuntu:

- Follow these steps
  - make sure you update it to the latest version (currently 13)

Login and setup postgres and postgres user:

1. Run the psql command from the postgres user account
   - `sudo -u postgres psql postgres`
2. Set the password
   - `\password postgres`
3. Enter a new password (pass@word1)
4. Close psql
   - `\q`

Run the following commands:

```
`sudo apt install make`

`sudo apt install gcc`

`sudo apt install postgresql-server-dev-13` (if the version ever changes, change that number at the end)
```

Setting up partman:

- In your working folder run:
  - git clone https://github.com/keithf4/pg_partman

- cd pg_partman
- make install

Commands that may come in handy:

`sudo service postgresql start` (starts postgres service)

`sudo service postgresql restart` (restarts postgres service)

`sudo service postgresql stop` (stops postgres service)

`explorer.exe .` (Use this if you fancy an explorer view instead of a terminal one. Also allows you to edit the files without using vim)

To get PgAdmin working:

Inside of the pg_hba.conf file, change the local connection to

`host    all             all             0.0.0.0/0            md5`

Inside of postgres.conf change listen_addresses under CONNECTIONS AND AUTHENTICATION to

`listen_addresses = '*'`

Example docs to get partman setup:

- How to

Additional Info

https://github.com/pgpartman/pg_partman/blob/master/doc/pg_partman.md

Partition types that pg_partman supports

- Time based
- Integer based

Peformance with 1 billion entries

Saved in the SPIKE:

- 38507 [Spike] Investigate using Partman (Postgres plugin) to partition audit logs by date
  Closed

How to run and Release Mobile

Debugging the iOS app:

1. Set efcMobileApp.iOS as the startup project
2. Pair the Mac by clicking this icon:
   image.png
   Image not found or type unknown
3. Make sure you are in Debug mode, not Release mode
4. Click the down arrow on the start options, and pick the device you want to simulate - Remote Device will not work
5. Happy testing!

Releasing the iOS app:

1. In the efcMobileApp.iOS/Info.plist file, scroll down to Bundle version and bundle versions string (short) then add 1 to each value.
2. Set efcMobileApp.iOS as the startup project
3. Pair the Mac (see step 2 of Debugging the iOS app)
4. Make sure you are in Release mode, not Debug mode
5. Right click on efcMobileApp.iOS and click Archive...
6. Wait for the Archive to finish - if there's a signing error:
   a. Right click on efcMobileApp.iOS and click Properties.
   b. Once there, click the iOS Bundle Signing tab and make sure the Signing Identity is: *iPhone Distribution: eFileCabinet, Inc.* and the Provisioning Profile is: *eFileCabinet Online AppStore* and try the Archive again.
   c. If there's still an issue, the signing certificate may be out of date. Please get with dev management to get a new signing certificate.
7. Once the Archive has finished, use VNC Viewer to connect to the Mac Mini (if you need credentials, ask dev management)
8. Open XCode
9. On the top toolbar, click Windows and then click Organizers
10. You should see Archives as a tab on the left, and in there you should see the archive you just created in Visual Studio
11. Select the archive and click distribute app
12. Follow the wizard, leaving all the defaults except the one asking about sending reports to Apple and XCode - uncheck that box
13. Please step through next steps with a member of the Dev management.
14. Login into https://developer.apple.com/ using login from Last pass: Mobile Dev Apple Account. (may need to be logged into the dev mac to get the authentication code on

login)

15. Go to the App Store Connect section and choose Apps:
image.png
Image not found or type unknown
16. Then go to the TestFlight tab at the top. You will now see the most recent version displayed. It will take a while to process.
17. Once done processing it will ask to complete some compliance information. Click on manage and then answer the questions as shown below (I had two different sets of questions I had to ask so both are listed below):image.png
Image not found or type unknown
image.png
Image not found or type unknown
image.png
Image not found or type unknown

image.png
Image not found or type unknown

image.png
Image not found or type unknown

18. Once you hit start internal testing make sure your apple ID is added to the internal testings users list and download the testflight app on your apple device. Once app is up and running log in and accept the invite to the app.

19. You are not good to test the app.

Debugging the Android app:

1. Set efcMobileApp.Droid as the startup project
2. Make sure you are in Debug mode, not Release mode
3. Run the emulator you have (you may need to install a new device)
4. Make sure you have enabled Hyper-V and Windows Hypervisor Platform in windows features, and that you have enabled Virtulization (or SVM on AMD) in your bios (If you don't android emulation probably won't actually work). You will want to restart your computer when you do this. Also see this doc for further guidance.
image.png
Image not found or type unknown

5. Wait for a long time - the Android emulator is very slow on startup
6. Alternative: If you have an Android phone, plug it in and on step 3 select your phone on the run dropdown. That way debugging should be a lot faster.

Releasing the Android app:

1. Release can not be done in Visual Studio 2022, Please use Visual Studio 2019
2. Make sure you've followed the steps found here to make sure JDK 11 is installed and will work with VS 2019.
3. Set efcMobileApp.Droid as the startup project
4. Make sure you are in Release mode, not Debug mode
5. Right click on efcMobileApp.Droid and click Archive...
6. Once the Archive has completed, click Open Distribution (you can also click the 'Distribute' button, and wak through through the steps, at the end just make sure you

upload to the lowest level track)
7. If at any point you are asked for signing credentials, they are found in the Android Signature Key Info file in the root of the source code.
8. If at any point you are asked for a keystore file, import the keystore file found in the root of the source code.
9. Give the resulting apk file to dev management to upload to the play store.

SonarCloud and SonarLint Setup Steps

Link your ADO account to SonarCloud

1. Go to the SonarCloud link and click "Log In" then "With Azure Devops" button
2. Let one of the senior devs know so they can add you to the SonarCloud project
3. Login again and make sure you can access the Utopia project in SonarCloud

Download and setup the SonarLint IDE tool

1. Go to https://marketplace.visualstudio.com/items?itemName=SonarSource.SonarLintforVisualStudio2019 and download the extension.
2. Restart VS if it was running. (Yes, you have to)
3. Inside VS go to Team Explorer > SonarQube > connect
4. You'll be prompted for a url and login for SonarCloud.

- Url is https://sonarcloud.io
- For username/password you'll need to generate an auth token. Go to SonarCloud, login, click on your profile pic > My Account. Now click the Security tab, enter a token name, and click Generate. I'd recommend saving this token into a password manager, you can only see it one time.
- Copy this token into the Username field back in the VS login prompt, no password is needed
- Select EFC Sonar Cloud Integration as the Project

5. In the Team Explorer > SonarQube tab, right click on Utopia and select Bind (Or update if it's already been bound)
6. It will take a minute, but all of the linter rules will be pulled down from the cloud and you should start seeing additional Warnings for security issues in the Error List tab on the bottom of the IDE (Error Code will start with an 'S')

How to Run WOPI Validator Tests on O365

The WOPI Validator Tool allows us to make sure our endpoints conform to the WOPI protocol so we can offer an Office 365 integration. This tool is run through Windows PowerShell, but in order to make it work, we need 4 things first:

1. The WOPI source code on our machine
2. A valid Rubex Authentication token
3. The port that we can use to run the WOPI tool with
4. The node id of the file we're running the WOPI too

Please note, number 4 is only needed if you're using a file endpoint from FilesController.cs in Utopia (wopi/files/{id}). There are other endpoints you can use, just be sure to update the powershell command accordingly. This is just an example using the file endpoint.

Source Code Setup (Utopia and WOPI)

1. Clone the WOPI source code onto your machine in a place easily accessible to you. A good option is just your source folder where your other repositories are.
2. Once the WOPI code is cloned, open the solution at the path: wopi-validator-core\WopiValidator.sln
3. Build the project in **Release** mode. This will add the necessary files and directories to enable you to run the tool in PowerShell later.
4. In Utopia, navigate to Utopia/Controllers/Wopi/WopiBaseController.cs - modify the ValidateRequestAsync method to always return true. We cannot run the proofkey portion of the test locally.

Rubex Authentication Token

1. Run Utopia
2. Go to the UtopiaDB and view the data in the DbUserSessions table.
3. Copy the token under the AccessToken column header on your current session row (i.e. 07680FAC-90D8-419B-AF81-5C14BFA23F4F). This token is the valid Rubex Auth Token you need to run the WOPI tool. Copy it and paste it somewhere that will be easily

accessible later.
    1. Alternatively, open up devtools networking tab in the Utopia window. Find an API call and copy the access token from the Authorization header
4. **Important: Keep your local Rubex session alive during the entire rest of this process or you will need to grab a new auth token!**

Port to Use

1. With the Utopia project still running, go to your taskbar and right click IIS Express.
2. Under the header "View Sites", you should see Utopia. Hover over it to see the ports Rubex is using.
image.png
Image not found or type unknown
3. There should be two ports, one that matches the port in the URL (i.e. localhost:54298) and one that's different (i.e. localhost:44313).
4. The port you want to copy and save for later is the one that **doesn't** match the URL port when you look at the browser running your Utopia project. The port you want is probably 44313 or 44311 or the like.
5. Keep track of it in the same place you saved your authentication token.

Node ID to Use

1. Download one of the wopi test files found here . It doesn't matter which one you use, there are just a few in here to try out.
2. Upload the file you chose to your localhost environment.
3. Once the upload completes, go to the UtopiaDB again and view the data in the DbNodes table.
4. Find the wopi test file you created and grab its Node ID from the Id column of the table (i.e. 294 or 11020). Keep track of it in the same place you saved your port and authentication token.

Putting it All Together

1. Open PowerShell in the wopi-validator-core\src\WopiValidator\bin\Release\net6.0\ directory
2. The first time you try to run a test with a new user, you must run the following test:
`dotnet Microsoft.Office.WopiValidator.dll -t TOKEN -l 0 -w https://localhost:PORT/wopi/files/NODE_ID -g PutUserInfo`
This will populate a string value on the user so that all subsequent tests will pass. If this is missing, you'll get error messages about missing the "UserInfo" property.
Substitute the TOKEN, PORT, and NODE_ID for your retrieved values.
3. If you want to run all of the WOPI Validations tests, run:
`dotnet Microsoft.Office.WopiValidator.dll -t TOKEN -l 0 -w https://localhost:PORT/wopi/files/NODE_ID --`

`ignore-skipped`
Substitute the TOKEN, PORT, and NODE_ID for your retrieved values.

4. If you want to run a specific Test Case of the WOPI Validation tests, run:
`dotnet Microsoft.Office.WopiValidator.dll -n NAME -w https://localhost:PORT/<YOUR ROUTE HERE> -t TOKEN -l 0`
Substitute the TOKEN and PORT with your retrieved values and the YOUR ROUTE HERE needs to match up to the endpoint you are testing. The NAME is where you place the name of the test case you want to run.
   - **For example:** If you are testing name `containers.EnumerateAncestorsOnRoot` then you should match it to the endpoint `/wopi/containers/NODE_ID/ancestry` where node is `NODE_ID` is your retrieved value from above.
   - Test case and test group names are all found inside of TestCases.xml found in the same folder as that .dll
5. See the **Additional Info** section for more commands

Troubleshooting

- If at any time, all your test cases are getting skipped and you want to see why, be sure to remove the --ignore-skipped flag from your command and run it again. If you see that the reason for skipping the test cases is a **306 Unused** error, that means your port is wrong. Make sure you're using one of the ports listed under Utopia in IIS Express, and make sure it's not the port your browser is using to connect to Rubex.
- If you are getting lots of Authorization exceptions while you're running your test cases, it simply means your Rubex Authentication token is no longer valid. Log out (if you're not already) and login again. Then follow the steps in the Rubex Authentication Token section of this document to get the new auth token. Replace the token in your command with this new one and run the test cases again.
- If you are seeing a message about missing the UserInfo property, see step 2 of "Putting it All Together"

Additional Info

[Wopi Documentation]

```
Microsoft.Office.WopiValidator 1.0.0
Copyright (C) 2018 Microsoft

  -w, --wopisrc          Required. WopiSrc URL for a wopitest file

  -t, --token            Required. WOPI access token

  -l, --token_ttl        Required. WOPI access token ttl
```

-c, --config (Default: runConfig.xml) Path to XML file with test definitions

-g, --testgroup Run only the tests in the specified group (cannot be used with testname)

-n, --testname Run only the test specified (cannot be used with testgroup)

-e, --testcategory (Default: All) Run only the tests in the specified category

-s, --ignore-skipped Don't output any info about skipped tests.

--help Display this help screen.

--version Display version information.

New work item

AWS DMS Setup

How to setup AWS DMS:

This is assuming the PostgreSQL DB is already created

SETUP SCHEMA:

A. Get the sqlserver2pgsql.pl script from

https://github.com/dalibo/sqlserver2pgsql/blob/master/INSTALL.md

- We will not be using Kettle

B. Install perl

- http://strawberryperl.com/

C. In MSSQ Studio, right click on the database required to migrate and select Task -> Generate Scripts

- Under SQL Server Management Studio, Right click on the database you want to export
- Select Tasks/Generate Scripts
- Click "Next" on the welcome screen (if it hasn't already been deactivated)
- Select your database
- In the list of things you can export, just change "Script Indexes" from False to True, then "Next"
- Select Tables then "Next"
- Select the tables you want to export (or select all), then "Next"
- Script to file, choose a filename, then "Next"
- Select unicode encoding (who knows..., maybe someone has put accents in objects names, or in comments)
- You will receive a SQL script (in SQL Server syntax) with Db Structure (tables, indexes, constraints, stored procedures, etc)

D. In the terminal, go to where the sqlserver2pgsql.pl script is kept and run the following:
`~/sqlserver2pgsql.pl -f <THE NAME OF THE FILE DUMP THAT CAME FROM PART C> -b before.sql -a after.sql -u unsure.sql -keep_identifier_case -stringtype_unspecified`

- The before script is what you should run before data migration
  - There may be some errors that come up when the perl script reads the MSSQ Db SQL dump. Mostly it's just syntax that can be manipulated inside of the dump.

    - If this comes up as an error it can be removed: 'ALTER TABLE [dbo].[DbNodeClosures] SET (LOCK_ESCALATION = DISABLE)'
    - You will probably have errors with the create views, this is just a whitespace issue. Remove that and you'll be golden.
    - It will probably ignore functions and procedures... Keep that in mind, it may not be important for procedures, and important for functions (although if you've already enabled cdc on tables, you can ignore those ignored functions)
- The after script is what you should run after data migration
    - keep in mind that postgres has a max constraint name size of 63 characters, so currently these will need to be renamed manually. View the alterations.sql file to see what I did (I had 48 alterations to adjust)
- The unsure script is what you should run after data migration but needs review and will more than likely need quite a few edits
    - Often it doesn't have views, functions, and other things
    - View the 'updated-unsure.sql' file as an example as it is what was used for the test migration

E. Connect to the PostgreSQL DB and run the before script (this will build the tables, types, and columns)

- I recommend using pgAdmin4
    - https://www.pgadmin.org/download/pgadmin-4-windows/
    - If you get an error having it run (eternal loading screen) then go into the Registry Editor -> \HKEY_CLASSES_ROOT.js -> Change 'Content Type' to text/javascript

F. Migrate your data over as listed in the 'SETUP AND RUN AWS DMS' section

G. Make sure that everything is migrated over

H. Connect to the PostgreSQL DB and run the after and unsure scripts
- Do so after you've reviewed them to make sure they're accurate and the constraint names are 63 characters or less

I. Go play some Starcraft, you've earned it

SETUP AND RUN AWS DMS:

1. Create your Replication Instance on AWS DMS
    - Not needed anymore, can just use the one that's already built
    - Make sure that the Replication Instance's IP address is allowed to enter the VPC security group of both the Source enpoint's DB and the Target endpoint's DB, if not, add it to both
2. Create your endpoints (source and target)
    - Test the connection with the Replication Instance before creating the endpoint (it will create even if the Replication instance can't connect to it)
3. Do everything here on the MSSQ DB

- https://aws.amazon.com/premiumsupport/knowledge-center/dms-migrate-rds-sqlserver/
- I know that it sucks to need to enable CDC for each table sooo just run the AutoWriter program if the tables are different then what's there, make sure you replace the strings in the file
- I've noticed better results (in that the task would run without errors) by just keeping 'exec sys.sp_cdc_stop_job;' and not turning it back on, though keep in mind that you probably won't be able to get your logs, so turn it back on if you need those
- You may get a table that fails, that's fine as it's probably a table that MSSQ made and isn't in the DB
  - Run these to get the tables if needed

```
For the dbo SCHEMA:
□□SELECT TABLE_NAME
□□FROM INFORMATION_SCHEMA.TABLES
□□WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA='dbo'
□□□
□For the workflow SCHEMA:
□□SELECT TABLE_NAME
□□FROM INFORMATION_SCHEMA.TABLES
□□WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA='workflow'
```

4. Create a DB migration task
   - Task Identifier: Something unique
   - Descriptive Amazon Resource Name (ARN): Skip this
   - Replication instance: Choose the Replication Instance from step 1
   - Select your Endpoints
   - Migration Type - I recommend 'Migrate existing data and replicate ongoing changes'
   - Editing mode: Wizard
   - Target table preparation mode: Do Nothing
   - Include LOB columns in replication: Full LOB mode
   - Maximum LOB size (KB): I set it to 10,000 but you can set it to whatever is needed
   - Enable validation: true
   - Enable CloudWatch logs: true
   - Advanced task settings
     - Create control table in target using schema: dms
     - Enable control table - enable all
   - Table mappings
     - Create two new selection rules and include SCHEMAs 'dbo' and 'workflow'
     - Create one new transformation rule with Target -> Schema, Schema name -> 'dbo', Action -> 'Rename to', Value -> 'public'
   - Migration task startup configuration: Automatically start
   - Save n start it

Previewer Setup for Local Environment

Use Docker To get the Previewer Up and Running: For local use

Taken from Accusoft's Try It link:

Accusoft: Try It!

1. Make sure you have stopped Prizm and Prizm Application Services so the ports won't conflict
2. Download Docker Desktop (Windows): Docker Desktop
3. Open up PowerShell and run this command:

```
docker pull accusoft/prizmdoc-viewer-eval:13.26
```

4. Find our Solution Name and License Key in LastPass under Prizm License and replace YOUR_SOLUTION_NAME and YOUR_LICENSE_KEY in the command below of your choice

- Option 1 (Recommended)-
  If you want to be able to restart the docker container use this (once this is run you only have to click stop or play in the Docker Desktop app):

```
docker run -p 8888:8888 -p 3000:3000 -p 18681:18681 -e ACCEPT_EULA=YES -e LICENSE_SOLUTION_NAME=YOUR_SOLUTION_NAME -e LICENSE_KEY=YOUR_LICENSE_KEY --name prizmdoc-viewer-eval docker pull accusoft/prizmdoc-viewer-eval:13.26
```

  - Once the PrizmDoc Server and PAS are running in PowerShell, and you can see the container named accusoft/prizmdoc-viewer-eval:latest in Docker Desktop you can close PowerShell. This will stop the container. Hit play in Docker Desktop to start it back up again.
- Option 2-
  If you want the docker container to be removed when you stop it use this:

```
docker run --rm -p 8888:8888 -p 3000:3000 -p 18681:18681 -e ACCEPT_EULA=YES -e LICENSE_SOLUTION_NAME=YOUR_SOLUTION_NAME -e LICENSE_KEY=YOUR_LICENSE_KEY --name prizmdoc-viewer-eval accusoft/prizmdoc-viewer-eval:13.26
```

5. It takes a few minutes for everything to start up, even though it says it's running in Docker. Confirm that everything is running with these checks:

- Option 1- Check that the sample application is running

  Sample Application
- Option 2- Check the server and service separately
  - Check PrizmDoc Server

    Health Check

    Admin Status
  - Check PrizmDoc Application Services(PAS)

    Health Check

    Service Connection

6. Once the application loads or all the checks come back with OK you are ready to start using the PrizmDoc Previewer with the AccusoftPreviewer solution.

Note: To get updates, just click the delete icon in Docker Desktop and repeat the steps above.

For debugging the front end of the Previewer:

On your machine, make sure you have node.js, npm, and gulp installed.

- Node needs to be a specific version. Check if node is installed by running `node -v`
  - If you have node version 11.3.0, skip to the npm installation
  - If you don't have node installed:
    - Use this msi: https://nodejs.org/download/release/v11.3.0/node-v11.3.0-x86.msi
    - Verify it installed correctly by running `node -v`
  - If you have a different version of node already installed, try installing nvm (node version manager) for Windows:
    - Download the nvm-setup.zip at this site: https://github.com/coreybutler/nvm-windows/releases , unzip it and run the application
    - Close out of your command line and reopen it (make sure to run it as an administrator)
    - Check that it installed correctly by running `nvm -v`
    - Run `nvm install 11.3.0` to install the correct version of node
    - Run `nvm use 11.3.0` to switch to the correct version
- Make sure npm is installed by running `npm -v`
  - If you don't have it installed, just run `sudo apt install npm` or `npm install -g npm`
    - If there are errors installing npm, make sure you update first by running `sudo apt-get update`
- To install gulp, simply run the command `npm install -global gulp`

Now you've got what you need. To run the previewer and debug it:

1. Run Utopia **and** Utopia\UtopiaPreviewers\AccusoftPreviewer at the same time.
2. In the source code, open the source code and go to the following directory: Utopia\UtopiaPreviewers\AccusoftPreviewer\AccusoftPreviewer\wwwroot\ViewerSources
3. Open a powershell window in that directory and run the command **./builddev.bat**
4. In the Utopia window, open the dev tools and make sure the cache is disabled
5. In the Utopia dev tools, you'll see a "Previewer" item in the tree view - if you expand it, you should see **webpack://**, which you can expand and find **./modules**
6. If you don't see **./modules** then you'll need to right click the Previewer in the browser and select **Reload Frame**. Then repeat steps 4 and 5.
7. In the ./modules directory, you'll see the client code for debugging.
8. Go eat a donut. You deserve it. You're ready to debug with the Previewer!

Troubleshooting

1. Make sure you have Azure Storage Emulator running.
2. If running the **./builddev.bat** script doesn't work, try running it again in PowerShell as an administrator
3. If that still doesn't work, try running the AccusoftPreviewer project as an administrator in Visual Studio.

The Way of Pain - To get the Previewer up and running:

1. Download and install the PrizmDoc Server and Client

- if the following steps dont't work to install the PrizmDoc Server and Client, see Quinn's comment
- https://v953w.app.goo.gl/ef1g
- For the user, the best thing to do is create a new system user and set a password to use for PrizmDoc. You CAN use your own user, but be aware -- if you change your password, PrizmDoc will not work until you update the password, see this link on how to update https://www.accusoft.com/faqs/is-it-possible-to-change-the-login-accounts-on-prizmdoc-services-in-windows/
- Also for user, if you created a new one, you may need to add the user to the Window's Administrators group.
- For the Server pick 'I have purchased a license' and then pick OEM. Then License info is the Development-Shared folder in Last Pass.
- Use the default URL for the client installer
- Other than the User/Password info, you should be able to just keep defaults on all options.

2. Restart your computer, yes... you have to.
3. Make sure the PrizmDoc server and client are running, (Demo isn't required to be running). These are services that need to be running called 'PrizmDoc' and 'PrizmApplicationServices'
4. Start the Utopia Project found Utopia\UtopiaPreviewers\AccusoftPreviewer - While this is running you can preview files in your local Utopia instance and the previewer project

webpage will be a HTTP Error 403.14 - Forbidden type page.

Note: Azure storage needs to be running or the previewer will timeout while loading

Note: Make sure your port number is correct in the AccusoftPreviewer.csproj file. The line <IISExpressSSLPort>44318</IISExpressSSLPort> got changed to <IISExpressSSLPort>44300</IISExpressSSLPort>. VS automatically sets this in some fashion, so make sure your hitting the correct port, I also made the mistake of trying to line it up to port 50560 (not the 44300 or the correct one 44318) which didn't help finding the problem. To not have problems in the future you need to update the ssl port in 3 different files.

- *C:\src\Utopia\UtopiaPreviewers\AccusoftPreviewer\AccusoftPreviewer\AccusoftPreviewer.csproj*
- *C:\src\Utopia\UtopiaPreviewers\AccusoftPreviewer\AccusoftPreviewer\AccusoftPreviewer.csproj.user*
- *C:\src\Utopia\UtopiaPreviewers\AccusoftPreviewer.vs\AccusoftPreviewer\config\applicationhost.config*

image.png
Image not found or type unknown

OCR Worker Local Setup

Keep in mind that these are the instructions that were used during testing of the OnPremise/Master merge that happened

1. Download Accusoft ImageGear (it is NOT a service)
2. Install it with the evaluators license
3. Inside of appsettings.json set "RunAsService" to false
4. Inside of App.config set
   `<add key="AccusoftSolutionName" value="EFileCabinet" />`
   `<add key="AccusoftSolutionKey" value="0x50DBF8B0,0x898D16D1,0x6BBB80AD,0xD99D98D8" />`
   to
   `<add key="AccusoftSolutionName" value="" />`
   `<add key="AccusoftSolutionKey" value="" />`
5. Inside of Program.cs inside of LoadWorkerConfiguration() comment out these two lines
   `AccusoftSolutionName = InstanceSettingsManager.Get<string>(InstanceSettingEnum.FileOCRWorkerAccusoftSolutionName),`
   `AccusoftSolutionKey = InstanceSettingsManager.Get<string>(InstanceSettingEnum.FileOCRWorkerAccusoftSolutionKey),`
6. Test that all is well by performing an OCR
   - You will probably get a GdPicture error, that is expected, just continue

Windows Services

Server Service:

First, we need to publish the Utopia Project.

image.png
Image not found or type unknown

After that we need to create the service by running the command (use Admin Command Prompt):
`SC CREATE "ServerService" binpath="C:\Projects\Efilecabinet\Utopia\Utopia\Utopia\bin\Release\netcoreapp3.1\publish\Utopia.exe"`

The bin path it's the path of the publish project + "Utopia.exe".

Now the service it's created. We can open the **Services** app of windows and check it.

image.png
Image not found or type unknown

The logs go to this path or similar:

image.png
Image not found or type unknown

Useful commands:

Delete the service:

`sc.exe delete "ServerService"`

Stop the service:

`sc stop "FileOCRService"`

List of the service names and projects:

FileOCRService - File OCR worker
Index Service - File Indexing worker
Rubex Service - Utopia Batch Worker
Server - Utopia

All the projects use .net core worker sdk, except File OCR Worker that continues on .net framework. For this project, we need to use the exact same name on the creation of the service. And if an error like this one ("*Login failed for user 'NT AUTHORITY\SYSTEM'. Reason: Failed to open the explicitly specified database 'UtopiaDB2'. [CLIENT: <local machine>]*") occurs, we need to create the user 'NT AUTHORITY\SYSTEM' on the db with all permissions.

Checking **Event Viewer** windows app it's useful for debugging services that fail to start for example.

image.png
Image not found or type unknown

Setup/Configurations for Mac Mini

Connecting

**You will need to be at our office or on the VPN to do this*

1. Download VNC Viewer
2. Make sure the Mac you're wanting to connect to is actually plugged in (check the ethernet cable) and turned on
   - The two QA ones are by all the other QA testing machines (as of 10/25/2022)
   - The Dev one is by McKay's space - in the Ops corner (as of 10/25/2022)
3. Once it's open, add a new connection see credentials
   - The dev mac currently is the only machine that has all the signing certificates for the Mac Desktop Client
     - The dev mac is often turned off
   - The QA mac 1 is much faster than any of the others, but doesn't have the signing certificates for the Desktop Client
     - When setting up the QA mac within VNC, turn Encryption > Prefer Off
     - Make sure no one from QA is using it

Credentials

DEV MAC

IP Address: 10.10.0.60
Login Info In Last Pass: Development Mac Mini

QA MAC 1

IP Address: 10.10.0.68
Login Info In Last Pass: QA Mac Mini 1

QA MAC 2 (not recommended)

IP Address: 10.10.0.67
Login Info In Last Pass: QA Mac Mini 2

Before you get started

You will need the following installed on your machine (Dev mac and QA mac 1 have these already):

- XCode
  - **Note**: Be careful with updating xCode versions, as it is likely to break building in release mode for the mac desktop client
  - Make sure you open XCode and check all the boxes to install the things needed
- Visual Studio for mac with:
  - .NET
  - .NET WebAssembly Build Tools
  - macOS (Cocoa) - *all versions listed*
  - Android
  - iOS
- Our codebase

Version Control

IMPORTANT - the master branch for the Mac Desktop Client is different from windows. You're looking for *origin/mac-master*. Branch off of that unless instructed otherwise

Sometimes git through Visual Studio for mac is really dumb, especially with changing branches. The good news is that you can use git through terminal where needed.

- If using Visual Studio for Mac for version control, make sure that when you bring the branch into your local environment, that you also track the branch, this will prevent a few issues

Useful Git Commands

- Force change branches => *git switch -f <branch-name>*
- It tries rebasing on branch swap sometimes => *git rebase --quit*

NuGet

You will probably need to add our UtopiaNuget at some point to a project on mac. The process is slightly different than what is described in our other wiki page.

1. Go to the NuGet packages in the solution and select Configure Sources

- image.png
  Image not found or type unknown
2. Go in to ADO and select Personal Access Tokens
   - image.png
     Image not found or type unknown
     1. Select + *New Token*
     2. Give it a name
     3. Change the expiration to be far in the future
     4. Give full access scope
     5. Click *Create*
     6. Save the token you are given and don't lose it
        - You will not be given it again and will need to regenerate another one if you lose it, potentially breaking other connections
3. Add a new package source
   1. Name it UtopiaNuget
   2. The location is https://pkgs.dev.azure.com/eFileCabinet/Utopia/_packaging/UtopiaNuget/nuget/v3/index.json
   3. The username is your ADO username
   4. The password is the token you generated in step 2
   5. Click *Add Source*
4. Click *OK*
5. At this point, if you select UtopiaNuget as the package source and are on the browse page, you should be able to see all of our packages we offer.

Troubleshooting

- If you finish all the steps successfully, and can see all the packages we offer, but restoring the packages still gives you a problem, then Visual Studio for mac messed up on adding a nuget reference to the global nuget file.
  - The fix is to search for the hidden file NuGet.Config and add the reference to the new package source manually
    - image.png
      Image not found or type unknown
    - image.png
      Image not found or type unknown (ignore the double reference)

Mac Desktop Client

IMPORTANT - the master branch for mac is different from windows. You're looking for *origin/mac-master*. Branch off of that unless instructed otherwise

If you're feeling courageous and want to mess with getting the signing certificates working on a machine other than the dev mac machine, then the credentials are found in last pass and is named Mobile Dev Apple Account.

Helpful Tips

It may be beneficial to run a local instance of Utopia to connect to for the Desktop Client.

- See our page on how to do this
- It is only possible to do from the office since you're remoting into the mac for testing

Building in Debug Mode

**If you're building on a machine that doesn't have the signing certificates (any other than Dev Mac Machine), you will need to set signing to false in the project file*.

To be able to build, you need to first have built two other projects on the machine:

- UtopiaAPIClient
- SharedLogic

It is very likely that you will need to update their references to UtopiaSharedClasses in NuGet. I recommend going into the UtopiaWindowsDesktopClient to update the packages and build them.

Building in Release Mode

*As of right now the signing certificates are only on the Dev Mac machine. I've tried putting it on the other mac, and they are there, but they require a password per use, which we don't have.

To build in release mode, it's as simple as putting it in release mode in visual studio, updating the version number in the Product.plist file, and running build. Once done, then go into the bin folder to get the dmg file

> *../UtopiaMacDesktopClient/RubexClient/bin/Release/RubexInstaller.dmg*

Send this to QA to test your changes. If they sign off on it then you're good to have it replace the dmg file in Utopia. Make sure you update the version number in the json file in the Utopia project after updating the dmg file!

Note

There is a chance that building it won't sign it properly or build the dmg. I never had this problem, but if you do then you will need to follow these steps:

> Monterey 12.0.1 has build issues when trying to build an installer from Visual Studios when running the release mode for Rubex Desktop Mac Application. Had to manually build the files. There is a Product.plist file in the obj/release folder that has a property called “os”, delete this entry and run the below commands (with the proper version).

```
productbuild --product /Users/dev/UtopiaClientApplications/UtopiaMacDesktopClient/RubexClient/obj/Release/Product.plist --component /Users/dev/UtopiaClientApplications/UtopiaMacDesktopClient/
```

```
RubexClient/bin/Release/Rubex.app /Applications --sign "Developer ID Installer: eFileCabinet, Inc. (J82X76G6Y8)" /Users/dev/UtopiaClientApplications/UtopiaMacDesktopClient/RubexClient/bin/Release/Rubex-1.0.4.pkg
```

```
appdmg /Users/dev/UtopiaClientApplications/UtopiaMacDesktopClient/RubexClient/InstallerItems/spec.json /Users/dev/UtopiaClientApplications/UtopiaMacDesktopClient/RubexClient/bin/Release/RubexInstaller.dmg
```

iOS Mobile

**I haven't touched this much but we do have a helpful page already built. I don't know how up to date it may be.*

Office Addins Terminal Services Environment

With Terminal Services, the Office Add-ins don't always show up, despite being installed for all users.
The batch script below imports the registry settings needed into HKEY_CURRENT_USER and can be run for each user to make the Office Add-ins show up. It is dependent on them installing it to the default path.

OfficeAddinRegistry.zip

Utopia Local DB Refresher

The Utopia local database refresher solution is located in the Utopia repo in the UtopiaLocalDatabaseRefresher folder.

1. Check that all the requirements are met before running:
    1. Make sure you've stopped the Microsoft Azure Storage Emulator
    2. PG Admin already set up with the local Postgres instance for Utopia
    3. Docker needs to be installed. (Docker Desktop is available on the Company Portal)
    4. Get ElasticSearch Running in Docker (see ElasticSearch Doc)
    5. DbMigratorEF has been rebuilt in debug (Open the Utopia solution, switch to DbMigratorEF and rebuild in debug)
    6. Rebuild UtopiaBatchWorker in debug as well
2. Open the UtopiaLocalDatabaseRefresher solution
3. (Optional) Update the "TestUserName" and the "TestUserPassword" to the desired values in appSettings.json
4. Ensure you have met the requirements
5. Run the solution in debug

Azurite Docker Doc

If things are broken and you need to do it **manually**:

- Make sure the Azure Storage Emulator or Azurite is running
- Make sure Elastic Search is running
- In pgAdmin4 Drop Cascade schemas: public and workflow
- Right click Schemas
    - Create a schema called “public”

- In the Utopia solution switch “Set startup project” to DbMigratorEF and run in debug (in command prompt press enter unless need other options)
- Switch back to Utopia project
- Run DBInstanceSettingsLocalConfig.sql from Utopia repo
- Open and run the UtopiaBatchWorker solution

Generating a DB Connection String

- Locate the ConfigurationDataCreator solution in the Utopia repo
- Start the solution

Create a new connection string

- Fill in the corresponding fields under Configuration Value
  - It can be helpful to click Load Configuration From System to see the local values
  - Most of this information can be found in LastPass or should be provided to you
  - Note that all fields are required to create a new connection string
- Click Create Configuration Data String
  - The new string will populate in the Configuration Data Output field

Pull information from a connection string

- Paste the encrypted connection string in the field Encrypted Value and click the Encrypt/Decrypt button
  - The decrypted value will be populated in the Decrypted Value field

Encrypt a connection string from a JSON object

- Paste the JSON object into the Decrypted Value field and click the Encrypt/Decrypt button
  - The encrypted value will be populated in the Encrypted Value field

Populate configuration values from a connection string

- Paste the encrypted connection string into the Configuration Data Input field and click Load Configuration Data String

How to Sign and Notarize a New Build for the Mac

I recently learned how to sign and notarize our build for the Mac Mini and will be documenting both the one-time setup that I had to do, as well as the steps that must be taken with each new build.

Many of these steps began as outlined in this blog, though with extensive changes along the way. If all you're looking for is how to make the next build for the Mac, please skip down to the "Notarize and Staple a Build" section.

Initial Setup

All of this setup has already occurred, and you probably won't ever need to do any of it again. These instructions are here just so that we can reproduce these actions if that is ever made necessary.

Update the Keychain certificates

In order to sign and notarize our build file, we're going to need to use two Keychain Certificates.

1. Open the Keychain Access app, open the "All Items" tab
2. Search for "J82X76G6Y8" (our "team identifier")
3. Among the results you are looking for these two certificates
   - Developer ID Application: eFilecabinet, Inc. (J82X76G6Y8)
   - Developer ID Installer: eFilecabinet, Inc. (J82X76G6Y8)
4. You need to verify that both of these are still current (not expired). You also need to make sure that there aren't any expired duplicates of these certificates. If there are, then right-click on the expired duplicates and delete them.

Configure Visual Studio to use the Hardened Runtime

For our application to be notarized, it is also necessary for it to be compiled using the Hardened Runtime.

1. Right-click on the RubexClient project in the Solution side-panel of Visual Studio and select Properties. Then open the Build->Mac Signing view
   - Check "Enable Hardened Runtime

- In the "Custom entitlements" field enter: Entitlements.plist
- Open the Entitlements.plist and add a necessary permission.
  - If you are doing this in Visual Studio, click "Add new entry" and then enter the following in each column
    - Property: com.apple.security.cs.allow-jit
    - Type: Boolean
    - Value: Yes
  - If you are doing this in a text editor, inside of the <dict></dict> section add the following:

    <key>com.apple.security.cs.allow-jit</key>
    <true/>

Configure Visual Studio to sign the .dmg build

Now Visual Studio needs to be configured to sign the package as part of the build process. For this you will need the name of the *Installer* certificate from the last step, as well as the identifier for our application (currently "com.efilecabinet.rubex").

1. In the Visual Studio project find and open the spec.json file
2. On the same level as the title, background, icon-size, contents, and window, add a new section as follows (with the current signing-identity and identifier as defined above):

   ```
   "code-sign": {
       "signing-identity": "Developer ID Application: eFilecabinet, Inc. (J82X76G6Y8)",
       "identifier": "com.efilecabinet.rubex"
   }
   ```

3. Next, right-click on the RubexClient project in the Solution side-panel of Visual Studio and select Properties. Then open the Build->Mac Signing view
   - Check "Sign the application bundle" at the top
   - For the "Identity" directly beneath that checkbox select "Developer ID: eFilecabinet, Inc. (J82X76G6Y8)"
   - Check "Sign the installer package" at the bottom
   - For the "Identity" directly beneath that checkbox select "Developer ID Installer: eFilecabinet, Inc. (J82X76G6Y8)"
4. Now when you build the project, towards the end of the process you should see a line that says:
   - [20/21] Signing image... [OK]

Generate Application-Specific Password for notarytool

Now that we have a signed build using the hardened runtime, we can configure our notarization tools. For this, we use the command-line utility "notarytool." First, though, we must authenticate the tool with an application-specific password. The current application-password is stored in LastPass, under the Notes section of the Mobile Dev Apple Account item. If you ever need to create

a new password, then:

1. Go to the Apple ID website
2. Enter the sign-in credentials under the Mobile Dev Apple Account item in LastPass
3. You will probably need to enter a second-factor code. For this you'll want to be signed into the Development Mac Mini so you can see the sign-in attempt alert, approve it, and get the 6-digit code to complete your sign in
4. Under the Sign-in and Security section select App-Specific Passwords
5. Press the + button, enter a memorable name, and press the Create button
6. You will *ONLY* be able to see the password one time on the next screen, so be sure to save that somewhere for reference

Authorize notarytool

Now we can use this password to authorize the notarytool.

1. In a terminal on the mac run:

   xcrun notarytool store-credentials --apple-id "mobiledev@efilecabinet.net" --team-id "J82X76G6Y8"

   Note the 10-digit team id at the end that we got from our certificates. Also, the Apple ID is the same one that we used to sign into the mobile dev account.
2. You will then be asked for a "Profile name." Choose something meaningful and write it down for future use. Our current profile name is "notary-credentials"
3. You will then be asked for the password that goes with your account (mobiledev@efilecabinet.net). This does *not* mean the password that you used to sign into the Apple Id account. It means the Application-Specific Password that you generated at the end of the last section.
4. It should process for a minute and then tell you that it has validated your credentials and saved them to Keychain. You will now be able to reference that Keychain item using the "Profile name" that you provided on step 2.

In the following section it refers to the keychain profile as "notary-credentials." Obviously, you would change that to whatever profile name you set in this section.

Notarize and Staple a Build

1. Increment the "Bundle versions string" in the Info.plist of the Visual Studio project
2. Ensure that your Visual Studio project is set to build for Release
3. Select Build->Rebuild Solution

4. Open a terminal and navigate to the build location: dev/UtopiaClientApplications/UtopiaMacDesktopClient/RubexClient/bin/Release/
5. Notarize the built .dmg file with the following command:

   xcrun notarytool submit RubexInstaller.dmg --keychain-profile "notary-credentials" --wait

   It will run for a while, just wait for it to complete with a success message.
6. Now you need to "staple" that notarization to the .dmg file. You do that with this command:

   xcrun stapler staple RubexInstaller.dmg

7. Finally, check that everything is correct by running this command

   spctl --assess -vv --type install RubexInstaller.dmg

   This should give a response message the .dmg file is "accepted," that the source is a "Notarized Developer ID," and the origin should be our developer id.

You may now give the signed and notarized RubexInstaller.dmg to QA for testing and then SecDevOps for release.

How to Renew a Certificate for Mac

To build and sign our Apple-product applications it is required for us to point the build towards valid certificates, which expire after a number of years. We will periodically need to renew these certificates. You can view the status of any certificates by opening Keychain Access. Any items that are already expired will have a red X by them, and every certificate's expiration date can be seen in a column to the right. To renew any expired/soon-to-be-expired certificates:

1. Open Xcode.
2. Don't open or create a project, just select Xcode->Settings from the top menu.
3. Select Accounts from the top ribbon menu.
4. Select the mobiledev@efilecabinet.net account (not mobiledev@efilecabinet.com). You may need to sign in again, using the credentials in LastPass.
5. Select the "eFileCabinet, Inc." team.
6. Click Manage Certificates.
7. Click the + dropdown button in the bottom-left.
8. Select the needed certificate type.
9. You should now see the new certificate in the main list, and it should have the same name as the previous certificate in that category.

****Note:** The next time that you try to build a project that relies on the new certificate you might get a popup requesting the "login" keychain password. This password should be the same one that you used to sign into the Dev account on the Mac machine (e.g. the password for the Development Mac Mini item in LastPass). Make sure to select "Allow Always" after entering the password so you won't have to do this again.

Go ahead and delete the old certificate once you verify that the new one is working for you. Having duplicates can make certain signing operations confused.

Local Testing

Test Utopia local changes in Atlantis local

1. Check that Utopia appsettings.json has:

```
"Environ": "Staging",
"Region": "LocalToStage"
```

2. In Utopia file SystemResourceController.cs search for AtlantisFrontendUrl = UtopiaSettings?.AtlantisFrontendUrl and change it to AtlantisFrontendUrl = "https://localhost:44437"
3. Make sure you're opted into Atlantis or in Utopia go to the file UserService.cs, search for the method GetAtlantisOptInStatusAsync, and have it return true
4. Check that Atlantis appsettings.json has:

```
"Environ": "Local"
```

Note: You will be using your local database. Don't change DbConnectionConfigurationData in Utopia appsettings.json.

Test Utopia local changes against local DB

1. Leave Utopia appsettings.json as:

```
"Environ": "Staging",
"Region": "LocalToStage"
```

2. Make sure you're opted out of Atlantis or in Utopia go to the file UserService.cs, search for the method GetAtlantisOptInStatusAsync, and have it return false

Test Utopia local changes against staging DB

1. Leave Utopia appsettings.json as:

```
"Environ": "Staging",
"Region": "LocalToStage"
```

2. Update DbConnectionConfigurationData to the staging connection string found in LastPass under "Utopia Staging DB AWS"
3. Make sure you're opted out of Atlantis or in Utopia go to the file UserService.cs, search for the method GetAtlantisOptInStatusAsync, and have it return false
4. If working remotely make sure you're on the VPN