This guide covers the set up and configuration of the AppDevPack for HCL Domino. In Chapter 1 we will discuss the set up for the Node.js implementation, in chapter 2 we will set up and configure the OAuth Service for Authentication (IAM). Both chapters include sample code to check the results and to get started.

Chapter 1: Set up the AppDevPack and domino-db.

In this first chapter we will install and set up the connection between Node.js and HCL Domino. This part consists of two components - the PROTON Server Add In Task for Domino which exposes parts of Dominos APIs vis gRPC and domino-db, the Node.js Module that HCL provides that wraps the gRPC compnents into a JavaScript API for use within Node.js.

We will install the PROTON Task. set up the initial communication and then add SSL Communications and a PKI based access to data. Finally, we will write some sample code to interact with domino-db, PROTON and HCL Domino to write data into an HCL Domino Application.

Step 1: write down your hostnames.

My Domino server cerrtifier is: CN=Proton1/O=C3UG/C=CA
My Domino host name is: proton1.c3ug.ca
My VM's hostname: proton1.fritz.box which is coming from my DHCP server (to find out your current hostname use:

hostnamectl status

on the command line. using *hostnamectl set-hostname <your-hostname.domain.qualifier>*

allows you to set/change your hostname if needed.

I also added the basic alias *proton1* to my hosts file in the CentOS VM to 127.0.0.1 as an additional alias as well as my Domino alias *proton1.c3ug.ca*

CAUTION: If you want to use Let’s Encrypt Certificates with IAM and not a Self Signed CA please make sure that this hostname can be reached via DNS services as Let’s Encrypt will try to call the host of the certificates using a DNS call. So your hostename should be publicly routable and reachable.

Step 2: Firewall Considerations

CentOS 7.x runs systemd firewall per default. using Webmin or other visual tools you can open up the ports needed by the various tools or do as I did and disable the VMs firewall as it is always running behind the Macs firewall plus the routers firewall. DO NOT DO THIS IN PRODUCTION - this is for dev/test purposes only !!!

To disable system3 use these commands as root:

```
systemctl disbale firewalld
```

and

```
systemctl stop firewalld
```

(to reverse that:

```
systemctl enable firewalld
```

and

```
systemctl start firewalld
```

to check the state use: `systemctl status firewalld`)

Step 3: Install Node.js

Now, to run the IAM server on top of the Domino server, we need to install Nodes.js. Currently (November 2019) IAM supports Nodejs V10.x which can be found here:
https://nodejs.org/en/

The link including the package name is:

https://nodejs.org/dist/v10.16.3/node-v10.16.3-linux-x64.tar.xz

Then, we have to install node manually as we can not use the latest stable release. I found this pretty helpful:

Install a Package from the Node Site

One option for installing Node.js on your server is to simply get the pre-built packages from the Node.js website and install them.
You can find the Linux binary packages here. Since CentOS 7 only comes in the 64-bit architecture, right click on the link under "Linux Binaries (.tar.gz)" labeled "64-bit". Select "Copy link address" or whatever similar option your browser provides.
On your server, change to your home directory and use the wget utility to download the files. Paste the URL you just copied as the argument for the command:

```
cd ~
wget https://nodejs.org/dist/v10.16.3/node-v10.16.3-linux-x64.tar.xz
```

Note: Your version number in the URL is likely to be different than the one above. Use the address you copied from the Node.js site rather than the specific URL provided in this guide.
Next, we will extract the binary package into our system's local package hierarchy with the tar command. The archive is packaged within a versioned directory, which we can get rid of by passing the --strip-components 1 option. We will specify the target directory of our command with the -C command:

```
sudo tar --strip-components 1 -xzvf node-v* -C /usr/local
```

-> Remark by Lothar Müller (edcom): in his version of CentOS (7.6) the syntax has changed so please try in case this is not working for you:

```
sudo tar --strip-components=1 -xzvf node-v* -C /usr/local
```

This will install all of the components within the /usr/local branch of your system.
You can verify that the installation was successful by asking Node for its version number:

```
node --version

v10.16.3
```

The installation was successful and you can now begin using Node.js on your CentOS 7 server.

This is the link where I shamelessly copied the above paragraph:

https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-a-centos-7-server

Step 4: Setup PROTON on the Domino Server

Stop your domino server, open a command line and switch to

```
/opt/ibm/domino/notes/latest/linux
```

make sure libnotes.so is present (`ls -l libnotes.so`)

expand the proton-addin archive (in my case from `/install)`

`sudo tar -xvf /install/proton-addin-<version>.tgz` ("TAB" is your friend here !)

fixup file permissons and ownership

```
sudo sh -v ./setup_proton.sh
```

start the Domino server

at the console you should see

```
PROTON> Build <version>-xxxxxxxxxxxxxxxxx
PROTON> Listening on 127.0.0.1, port 38770, INSECURE
PROTON> Note: Requested port was 0, Actual listen port is 38770
PROTON> Server initialized
PROTON> Server only allows Anonymous access.
```

if not, issue a "`load proton`" command and check outputs

if necessary, add the PROTON task to the SERVERTASKS= line in the notes.ini in /local/proton/domino-data/notes.ini in my case.

The proton task should now be up and running

Step 5: Securing PROTON with SSL and Client Authentication

Step 5.1: Downloading and installing the KYR-Tool

If you ever dealt with SSL and Domino you might be familiar with the KYR-Tool. If not, this is a special tool to package SSL Certificates in a keyring file in a way that Domino can consume. This tool is not be default part of the domino server install and has to be downloaded seperately. Thankfully, HCL Domino V10 and higher already have kyrtool installed - pleae check in

```
/opt/ibm/domino/notes/latest/linux
```

for it. If it's not there yet, you can find the download here:

https://www-945.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~Lotus&product=ibm/Lotus/Lotus+Domino&release=9.0.1.2&platform=All&function=fixId&fixids=KYRTool_9x_ClientServer

You will need the Linux 64 bit version and have your IBM-ID at hand :-)

the resulting file "kyrtool" must be placed into `/opt/ibm/domino/notes/latest/linux/`

Step 5.2: Creating the certificates that are needed.

In this tutorial, we will focus on self-signed SSL certificates and the scripts that are provided therefor by IBM/HCL in the appdev pack.

First, we have to create the base Certification Authority (CA) and the CRT and KEY files for the proton server and the technical user.

This is done by running the script as root like so:

```
./make_certs.sh
```

Of course, we have to make some changes to this script before running it - it has to be tailored to your domino server.

using nano, edit the script. Here are my changes (at the top and the bottom section of the file)

top:

```
if [[ -f ca.key || -f ca.crt ]]
then
      echo "CA already exists."
else
      # Generate CA private key
   (set -x ; openssl genrsa -passout pass:1234 -des3 -out ca.key 4096)
   # Self-Sign CA key
   (set -x ; openssl req -passin pass:1234 -new -x509 -days 365 -key ca.key
-out ca.crt -subj "/O=C3UG/CN=C3UG-Test-CA" -sha256)
```

These changes are related to my certifiers in Domino. I would think they are not needed but I had issues importing the client certs later for the technical users without them. Might as well be only me but this worked for me. **SO PLEASE USE YOUR CERTIFIER STRINGS here accordingly !**

bottom:

```
create_key server "/O=C3UG/CN=PROTON1" "DNS:proton1.fritz.box"
create_key app1 "/O=C3UG/CN=app1" ""
create_key app2 "/O=C3UG/CN=app2" ""
```

After some talks on slack with Oliver Busse, please make sure that your file access to the Domino paths is correct for the notes user. Sometimes even running the scripts using sudo gives you errors with certain files, e.g. server.key. In this case, check your permissions, delete the old stuff from a previous run and try again. Running as root is an option but you will have to chown your user right after running the scripts for the notes user or you have the next pitfall. Don't worry, running those scripts does not do anything harmful to your server environment so you can run them until the result is ok. Patience is of the essence here.

This will create "server.key, server.crt, app1.crt, app1.key, app2.crt, app2.key, ca.crt and ca.key, etc.)

Step 5.3: Create the Domino Keyring file

As of AppDevPack 1.0.2, the following script updates are no longer necessary.

For AppDevPack < 1.0.2:

In a great effort, Oliver Busse fixed the scripts, so download `make_keyring.sh` from here:

https://gist.github.com/zeromancer1972/74ddbdc655bf15616cdc1928d522730b

and copy them to `/opt/ibm/domino/notes/latest/linux/`

----

using nano, edit the kyrtool function of your make_keyring.sh file to point to your notesdata folder, in my case `/local/proton/notesdata`

I also recommend to rename the keyring and the stashfile names "sample1.kyr" and "sample1.sth" into something more meaningfull like your servername, in my case proton1.kyr and proton1.sth. Also. You can think about changing the target directory here which is “tmp”. If you want to place the files someplace else, enter the desired path here. For the next step, I am assuming you will create the files in the “tmp” directory.

running this script, you will end up having 2 new files in /tmp:

- proton1.kyr
- proton1.sth

copy those files to /opt/local/proton/dominodata

Step 5.4: Setting up SSL for PROTON

Stop your domino server.

Got to `/local/proton/dominodata` (your data directory that is)

edit notes.ini using nano.

Add the following lines to your notes.ini if not yet there:

```
PROTON_LISTEN_ADDRESS=0.0.0.0
PROTON_LISTEN_PORT=3002
PROTON_SSL=1
PROTON_KEYFILE=proton1.kyr
```

restart the domino server, PROTON TASK should start saying secure and anonymous access only.

Step 5.5: Create technical user and import client certificate

refer to this section here:

Client authentication

Proton authenticates client application requests based on the setting of the PROTON_AUTHENTICATIONnotes.ini setting. Valid options are:

- client_cert: The client certificate is mapped to a Person document in the server's directory. Access to data is calculated based on this identity. Proton must be enabled for TLS/SSL for this option.
- anonymous: All requests are made as the Anonymous user identity. This name does not need to appear in the directory, but it does need to exist in database ACL. This option is available with and with out TLS/SSL being enabled.

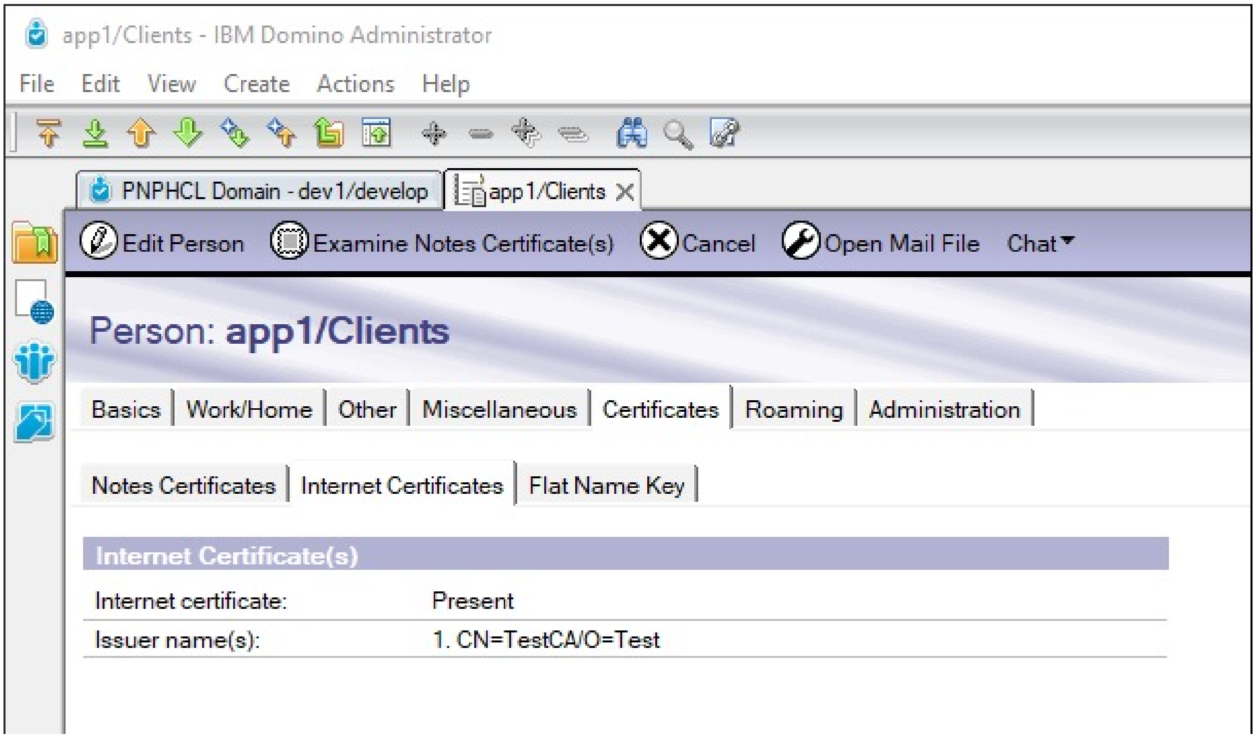
The default behavior when the setting does not exist is to provide Anonymous access to Domino databases.

client-certificate-authentication

To require that applications provide a valid client certificate set the following notes.ini variable:
PROTON_AUTHENTICATION=client_cert
With this setting enabled there are some additional administrative and client requirements.

I. The client application must supply a valid client certificate when making domino-db requests to the Proton server on Domino. The common name in the client certificate must have a name that can found in the Domino directory. Proton performs a lookup in the Domino directory to find the person document.

II. The Domino administrator must create a Person document in the Domino directory and perform the Import Internet Certificates Action on the Person document. This is required because the client certificate is verified against the known certificate in the Domino directory.

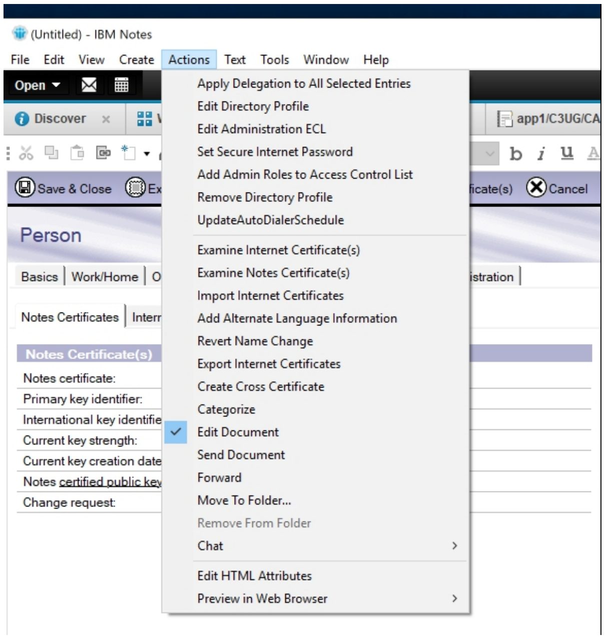


Internet certificate: Present

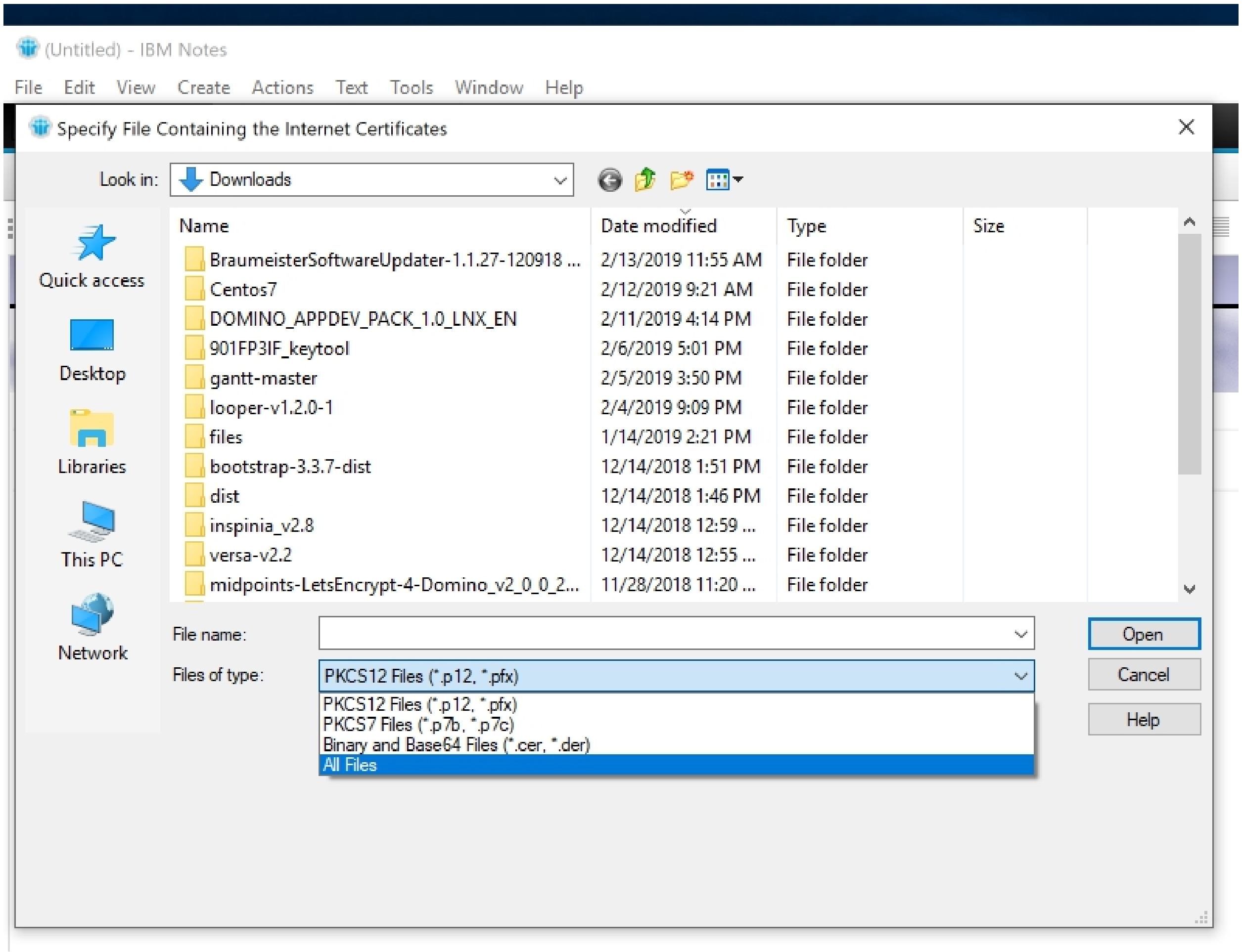
Issuer name(s): 1. CN=TestCA/O=Test

Now - a lot of people get stuck here - import the certificates by importing the app1.crt file. No need to create a *.pem or whatever.
Example using app2, please use app1 for your demo:

Create a person record for app1. Then after saving it, select the Actions Menu...



and select "Import Internet Certificates" ! This brings up the following dialog:



(Untitled) - IBM Notes
File Edit View Create Actions Text Tools Window Help
Specify File Containing the Internet Certificates
Quick access
Desktop
Libraries
This PC
Network
Open
Cancel
Help

Look in: Downloads

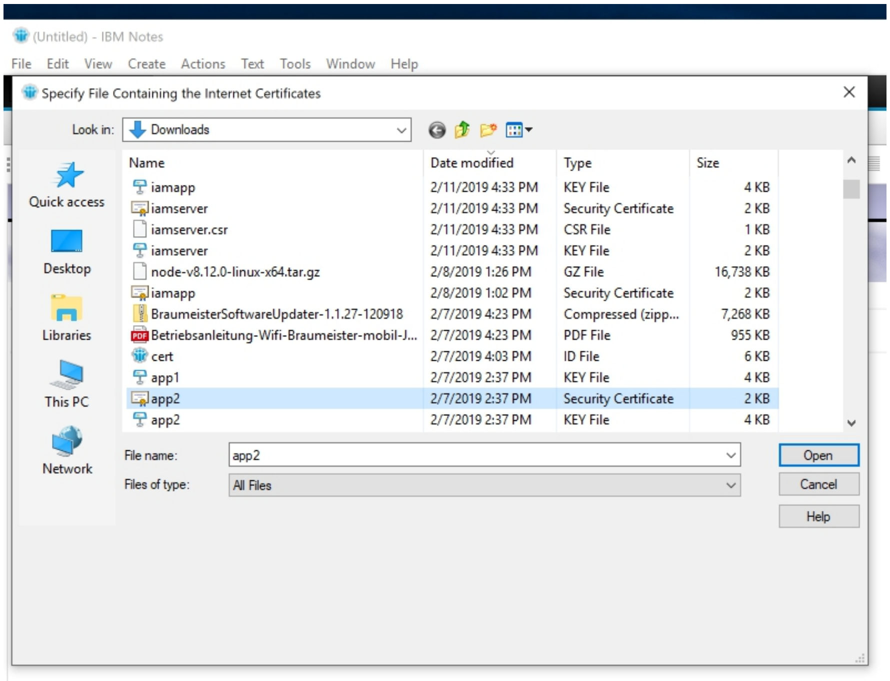
Name	Date modified	Type	Size
BraumeisterSoftwareUpdater-1.1.27-120918 ...	2/13/2019 11:55 AM	File folder	
Centos7	2/12/2019 9:21 AM	File folder	
DOMINO_APPDEV_PACK_1.0_LNX_EN	2/11/2019 4:14 PM	File folder	
901FP3IF_keytool	2/6/2019 5:01 PM	File folder	
gantt-master	2/5/2019 3:50 PM	File folder	
looper-v1.2.0-1	2/4/2019 9:09 PM	File folder	
files	1/14/2019 2:21 PM	File folder	
bootstrap-3.3.7-dist	12/14/2018 1:51 PM	File folder	
dist	12/14/2018 1:46 PM	File folder	
inspinia_v2.8	12/14/2018 12:59 ...	File folder	
versa-v2.2	12/14/2018 12:55 ...	File folder	
midpoints-LetsEncrypt-4-Domino_v2_0_2...	11/28/2018 11:20 ...	File folder	

File name:

Files of type: PKCS12 Files (*.p12, *.pfx)

PKCS12 Files (*.p12, *.pfx)
PKCS7 Files (*.p7b, *.p7c)
Binary and Base64 Files (*.cer, *.der)
All Files

Select "All Files" an pick the `app1.crt` file !



Specify File Containing the Internet Certificates

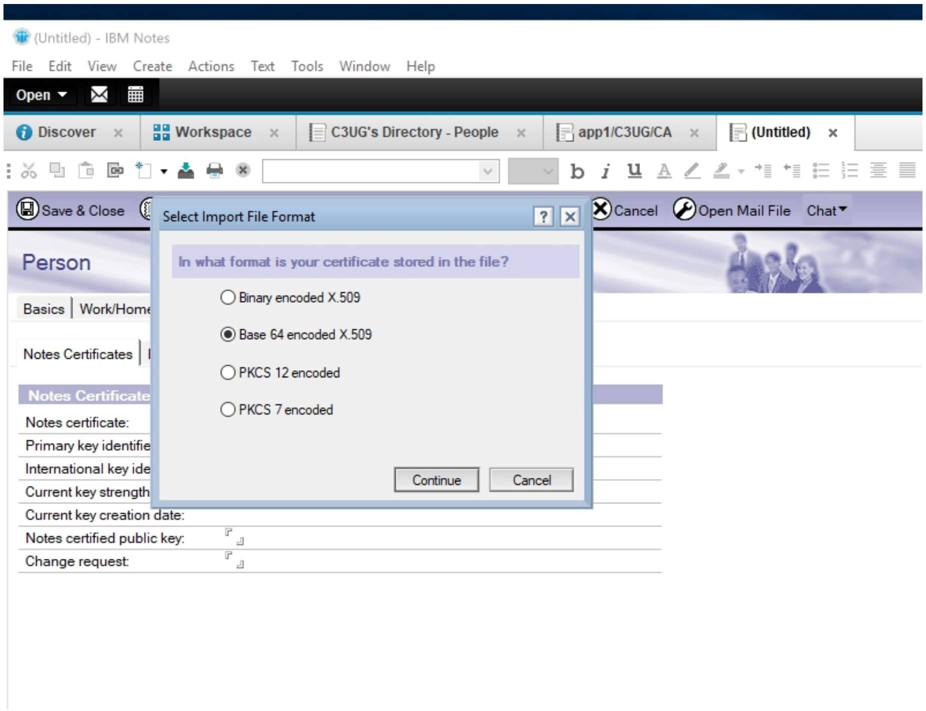
Look in: Downloads

Name	Date modified	Type	Size
iamapp	2/11/2019 4:33 PM	KEY File	4 KB
iamserver	2/11/2019 4:33 PM	Security Certificate	2 KB
iamserver.csr	2/11/2019 4:33 PM	CSR File	1 KB
iamserver	2/11/2019 4:33 PM	KEY File	2 KB
node-v8.12.0-linux-x64.tar.gz	2/8/2019 1:26 PM	GZ File	16,738 KB
iamapp	2/8/2019 1:02 PM	Security Certificate	2 KB
BraumeisterSoftwareUpdater-1.1.27-120918	2/7/2019 4:23 PM	Compressed (zipp...	7,268 KB
Betriebsanleitung-Wifi-Braumeister-mobil-J...	2/7/2019 4:23 PM	PDF File	955 KB
cert	2/7/2019 4:03 PM	ID File	6 KB
app1	2/7/2019 2:37 PM	KEY File	4 KB
app2	2/7/2019 2:37 PM	Security Certificate	2 KB
app2	2/7/2019 2:37 PM	KEY File	4 KB

File name: app2

Files of type: All Files

use `app1.crt` here !



Notes Certificate

Notes certificate:

Primary key identifie

International key ide

Current key strength

Current key creation date:

Notes certified public key:

Change request:

Select Import File Format

In what format is your certificate stored in the file?

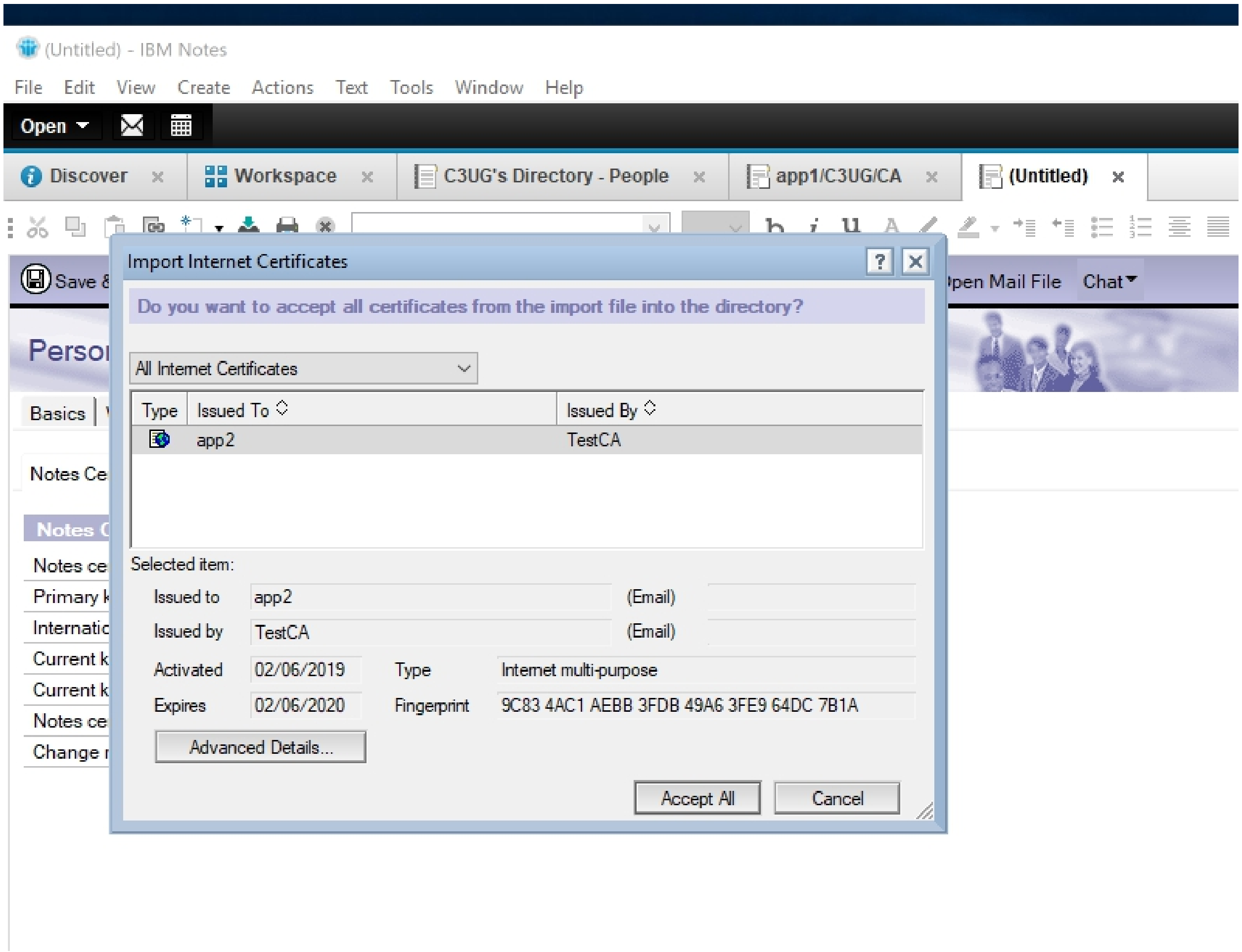
○ Binary encoded X.509

● Base 64 encoded X.509

○ PKCS 12 encoded

○ PKCS 7 encoded

keep the format as is.



Import Internet Certificates

Do you want to accept all certificates from the import file into the directory?

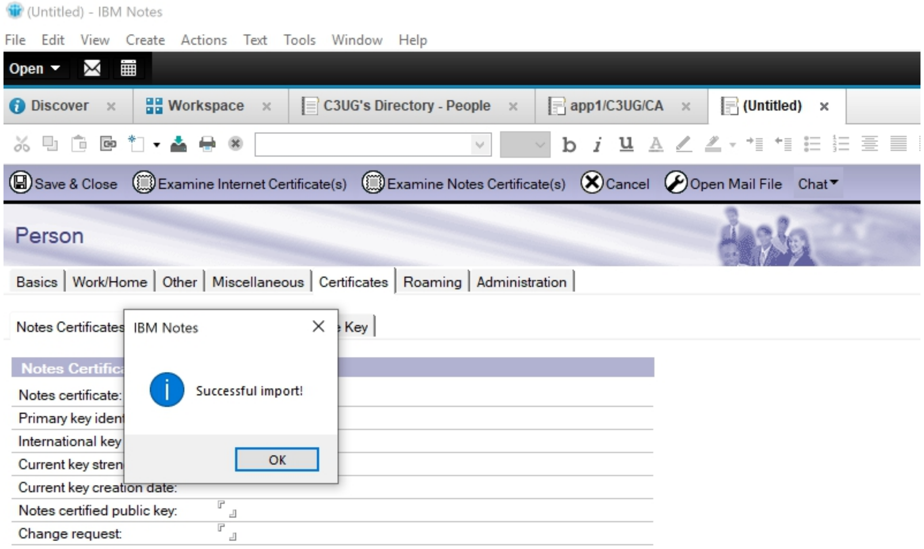
All Internet Certificates

Type	Issued To	Issued By
	app2	TestCA

Selected item:

Issued to	app2	(Email)	
Issued by	TestCA	(Email)	
Activated	02/06/2019	Type	Internet multi-purpose
Expires	02/06/2020	Fingerprint	9C83 4AC1 AEBB 3FDB 49A6 3FE9 64DC 7B1A

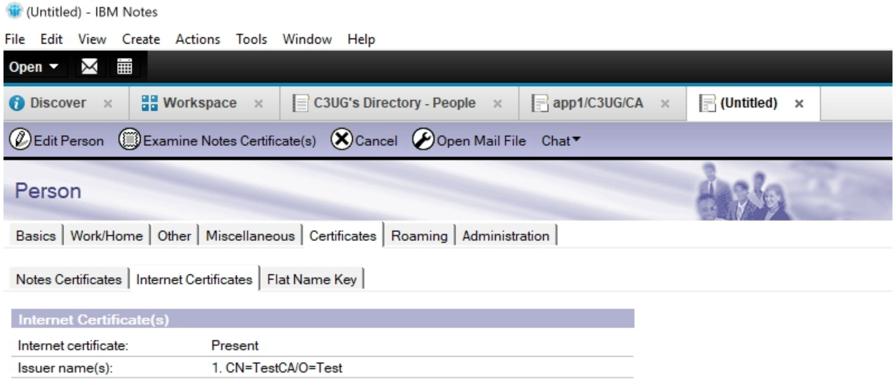
Click "accept all"


(Untitled) - IBM Notes
File Edit View Create Actions Text Tools Window Help
Open
Discover
Workspace
C3UG's Directory - People
app1/C3UG/CA
(Untitled)
Save & Close
Examine Internet Certificate(s)
Examine Notes Certificate(s)
Cancel
Open Mail File
Chat
Person
Basics | Work/Home | Other | Miscellaneous | Certificates | Roaming | Administration
Notes Certificates
Key
Notes Certific
Notes certificate:
Primary key ident
International key
Current key stren
Current key creation date:
Notes certified public key:
Change request:
IBM Notes
OK

Successful import!

Save & Close and re-check:

Done !


(Untitled) - IBM Notes
File Edit View Create Actions Tools Window Help
Open
Discover
Workspace
C3UG's Directory - People
app1/C3UG/CA
(Untitled)
Edit Person
Examine Notes Certificate(s)
Cancel
Open Mail File
Chat
Person
Basics | Work/Home | Other | Miscellaneous | Certificates | Roaming | Administration
Notes Certificates | Internet Certificates | Flat Name Key
Internet Certificate(s)

Internet certificate: Present

Issuer name(s): 1. CN=TestCA/O=Test

Beginning with AppDevPack 1.0.2, you can also import the client certificate via the server console:

```
load proton --importcrt app1.crt
```

Step 5.6 Add Client Authentication to PROTON

Stop the domino server, edit notes.ini

add the following line to your Notes.ini:

```
PROTON_AUTHENTICATION=client_cert
```

restart your server.

Now, this concludes the SSL/Authentication part !

Step 6: Test your configuration

Now, let's start coding a bit.

The appdev pack comes with samples that let you test the connectivity using SSL and Authentication:

We will use the domino-db Quick start sample. Inside of the AppDevPack install package, there's a sample Notes Application called node-demo.nsf. Please copy this database to your HCL Domino server running the PROTON Task as installed previously. Set the ACL accordingly (put you technical users in and give them for example editor access). Please sign the database ith an admin or server id.

This sample will create two new documents in the sample database using the technical user and SSL commuication as installed so far.

Prerequisites:

- Node.js and npm need to be installed. If you want to install it on a linux machine, see the install guidelines above. If not, check your operating systems install guides. Again, make sure to install the supported LTS Version 10.x - currently, the AppDevPack 1.0.2 will not work with LTS 12.x yet
- You will need the domino-db archive file (domino-domino-db-1.x.x.tgz) from the AppDevPack install package.
- I recommend installing MS Visual Studio Code on your machine for editing purposes. Follow the instructions for your operating system here: https://code.visualstudio.com/Download

We will start by creating an application in a new folder, we will call it `my-app`. Create the folder and continue doing:

Caution ! As of writing, this relates to domino-db Version 1.0.2. Please replace the numbers below according to your version !

```
cd ../my-app
npm install <path to domino-domino-db-1.x.x.tgz>/domino-domino-1.3.0.tgz -
-save
```

Now, inside of `/my-apps`, create a sub folder called `certifcates` and copy the following files created previously in the install process of the AppDevPack into this folder:

```
/my-apps/certificates
```

- app1.crt
- app1.key
- ca.crt

Start your code editor and create a file named `index.js` in the `/my-app` folder like so in case you are using Visual Studio Code:

```
code .
```

Next, create a new file called `server-config.js` in the `/my-app` directory.

copy the following code into `server-config.js`:

```
var fs = require('fs');
var path = require('path');
const readFile = fileName => {
      try {
            return fs.readFileSync(path.resolve(fileName));
      } catch (error) {
            console.log(error);
            return undefined;
      }
};
const rootCertificate = readFile('./certificates/ca.crt');
const clientCertificate = readFile('./certificates/app1.crt');
const clientKey = readFile('./certificates/app1.key');
const serverConfig = {
      hostName: '<your proton server hostname>', // DNS (!) Host name of your
server
// See scripts to create kyr-file and ca for adoption !
      connection: {
            port: '3002', // Proton port on your server
            secure: true,
      },
      credentials: {
            rootCertificate,
            clientCertificate,
            clientKey
      }
};
module.exports = serverConfig;
```

Make sure to change the “Hostename” variable to your DNS name of your HCL Domino Server running the proton task !

now, we create a new file called “`index.js`” inside of `/my-apps`

Now, copy the following code into `index.js`:

```
const { useServer } = require('@domino/domino-db');
const serverConfig = require('./server-config');

const databaseConfig = { filePath: 'node-demo.nsf',} // The database file name
};
const createOptions = {
      documents:
            [
                  {
                        Form: 'Contact',
                        FirstName: 'Aaron',
                        LastName: 'Aardman',
                        City: 'Arlington',
                        State: 'MA',
                  },
                  {
                        Form: 'Contact',
                        FirstName: 'Brian',
                        LastName: 'Zelnick',
                        City: 'Chelmsford',
                        State: 'MA',
                  },
            ],
      };
useServer(serverConfig).then(async server => {
      const database = await server.useDatabase(databaseConfig);
      const response = await database.bulkCreateDocuments(createOptions);
// Display the new document UNIDs
      const unids = response.documents.map(doc => doc['@unid']);
      console.log(`Documents created: ${unids}`);
});
```

Save all of the code and exit your code editor to the command line.

Start the sample app like so:

```
npm run index.js
```

You should see an output similar to this on the console:

```
node index.js
Documents created:
81AE9D4D33124565842584D2004786A0,577D15F401C88DF3842584D2004786A5
```

This concludes the first chapter to set up domino-db !

Chapter 2: Set up IAM as OAUTH authentication service for Domino and domino-db.

This part of the tutorial consists of the following pieces:

- Create public facing SSL Certificates using Lets Encrypt and Certbot
- Create the necessary SSL Certificates for IAM and the parts needed for PROTON
- Prepare the Domino Server
  - Set up a Credential Store if not present
  - Set up ID Vault if not present
- Set up IAM step-by-step
- Set up Act-As-User Feature
- Set up DSAPI Filter for OAuth usage with Domino HTTP Server

Set up IAM Server using Let’s Encrypt Certificates for public facing SSL certificates

This next passage is copied from Daniel Nashed’s Blog and has some slight changes for this tutorial. Here’s the link to Daniel’s original article:

http://blog.nashcom.de/nashcomblog.nsf/dx/appdevpack-security-setup-explained.htm?opendocument&comments#anc1

---- Daniel Start ----

Overview of Components

The AppDevPack consists for the following main parts:

Proton Server task

This is the heart of the AppDevPack and implements the new GRPC protocol.
Access to this task can be either anonymous, by client certificate or new with the IAM service with an OAUTH2 token.
Proton is a server task which needs a keyring file with a proper certificate. The certificate needs to be issued to the DNS name of the server (for example proton.nashcom.loc).

The communication with the GRPC protocol (e.g. DQL queries) are handed by this server task.
Proton hasn't got a well known port today. Choose any port above 1024 (the ports below are restricted ports on Linux).
I have chosen 1353 on my server. In this tutorial. we will be using port 3002 but you can change that to your needs.

Notes Database "iam-store"

The database is used to store IAM data on the server. This database has to be created from **iam-store.ntf** which isn't in the Proton package but in the IAM server package.
During installation you have to copy it from the IAM server package.
The IAM Client user.id needs to have editor access to the database, because the IAM server will use the client certificate to authenticate on the Proton server and access the database via Proton.

DSAPI Filter liboauth-dsapi.so

The DSAPI filter is an extension to the HTTP task and is needed to allow authentication for additional services implemented via HTTPS (e.g. Calendar services)

If you want to use those services you have to properly configure HTTPS (with another keyring file) to allow secure communication.
This is really a separate component which is independent from Proton.
The DSAPI filter confused me, when first looking into this. But it is a separate part of the setup with a separate access to Domino via HTTPS beside Proton access.
This DSAPI will communicate with the IAM server to verify tickets and stores tickets also in the Web credential store.

Helper Server task oauthcfg

This server task is needed to configure the trust store for the DSAPI filter and is just invoked for configuration.
It is executed once to configure an additional configuration and writes the credential store configuration for the DSAPI Filter.
Before you can do that, you need to configure the credential store first.

IAM Server (Node.js application)

IAM stands for **I**dentity and **A**ccess **M**anagement. This is a standard term which is also used by different products
(see for example: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html).

It is used to log in users and also to control access to applications.
This service is used for GRPC and also for HTTPS requests.
IAM leverages LDAPS communication for authenticating users and speaks on it's own with the Proton server task for example to safely store the user tickets in a central Notes database iam-store.

In most cases the user data is either in Domino Directory or in Active Directory in most of the cases. And we need a LDAPS with a proper certificate and matching name for the LDAP Server (it could be the Proton server).

IAM Client User ID

The IAM server stores it's data securely in **iam-store.nsf**.
Beside a client certificate for communicating with Proton, the IAM server needs a person document and **Notes.ID** which is stored in **ID-Vault** to be able to encrypt IAM data on the server.
The certificate is read from ID Vault and used for encryption.

Overview of Certificates and Keys

For secure communication all those components need to be able to communicate over SSL/TLS.
This requires Server and Client X.509 certificates. Let me describe the required certificates below.
The overview should help you to understand how all components work together from the security side.

Notes.ID for IAM Client / X.509 Client Certificate/Key

First of all you need a Notes.ID for the IAM Service. The IAM Client component of the IAM server uses a client certificate to communicate with the Proton server.
Beside the client certificate this user needs to a Notes.ID which is also stored in an ID-Vault!
The Notes.ID is used for encryption. The client certificate is used for authentication with the Proton server.
You need to import the client certificate (X.509) into the person document.
On the other side it is stored on the IAM server. The certificate is stored in PEM format and needs a password, which needs to be configured on the IAM server.

Server Certificate for Domino Server

Any Domino server accessed via HTTPS needs a proper Web Server Certificate which is stored in a keyring file.
Encrypted communication is required for HTTPS (port 443) access and also for LDAPS (Secure LDAP communication over port 636).
This could be on the same server or two different servers. In that case you would need two separate Domino keyring files with different host names.

Server Certificate for Proton

Proton uses the keyring file format to store the server key and certificate. It also needs a Web Server Certificate for secure communication.
The keyring file needs to contain the trusted root for their own certificates and also for the IAM Client certificate.
In the best case all your certificates are issued by the same CA. That would not require to add trusted roots and intermediate certificates into the keyring file.

Server Certificate for the IAM Server

Also the IAM server needs a Web Server certificate for secure communication.
This is a standard Web Server certificate and stored in PEM files instead of a keyring file on the Domino server side.

Client Certificate IAM Client

The IAM server needs to communicate with the Proton Server tasks.
Client certificate based authentication needs a X.509 certificate which is trusted by the Proton keyring file for the issuing CA.
The person document for the IAM Client needs to contain the Internet (X.509)Certificate.
You can manually import the file into the person document. We will show this later in this tutorial or please check the “Setup Proton” tutorial for details.

Summary: List of Certificates used

- Web Server Certificate (keyring) for Domino HTTPS and LDAPS
- Web Server Certificate for Proton (keyring)
- Web Server Certificate for IAM Server (PEM file)
- IAM Client Certificate (PEM password protected)
- IAM Notes ID with X.509 Client cert added to the person doc. Notes.ID must be available in ID Vault!

---- Daniel End----

What goes where ?

The AppDevPack components get extracted into your Domino server files directory which (for V10) is:

```
/opt/ibm/domino/notes/latest/linux
```

and will be since V11:

```
/opt/hcl/domino/notes/latest/linux
```

The KYRTOOL that is needed to build Domino Keyring Files comes preinstalled with V10 and is also there:

```
/opt/ibm/domino/notes/latest/linux/KYRTOOL
```

Your certificates to work with Proton and Domino go to

```
/local/notesdata
```

You will need access to your Proton KYR-File, so make a note of its name here: ___________________

Should be in your notes.ini: `PROTON_KEYFILE=<your-kyr-file>.kyr`

The IAM Service will be deployed/extracted to a Directory the Notes User has access to. In my sample I used:

```
/home/notes/
```

and created a directory there:

```
mkdir iam-service
```

Here, I transferred the files from the package `domino-iam-service-1.2.5` from the AppDevPack Package. I ssh'd as the Notes user into my VM. If you use any other user to do the transfer, make sure to change ownership in the linux file system to the notes user.

Configuring the SSL Certificates with the local CA created for the PROTON SSL Config (Self Signed) using OpenSSL

Do this in the subdirectory "iam-service" as described above. You will need you ca.crt, ca.seq, ca.key files from the proton ssl setup. Copy them to this directory.

Step 1: create a file called iamserver.cnf that includes the following:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name (full name)
localityName = Locality Name (eg, city)
organizationName = Organization Name (eg, company)
commonName = Common Name (e.g. fully qualified host name)

[ req_ext ]
subjectAltName = @alt_names

[alt_names]
DNS.1 = iamserver.com
```

Make sure the hostename of the IAM Server is in the alt_names list. During the CSR file creation, you are asked to input the passphrase of the private key and the certificate information.

Step 1: Create a Certificate Signing Request (CSR) for the IAM Server:

```
openssl req -new -out iamserver.csr -key iamserver.key -nodes -config iam-
server.cnf
```

This creates the files iamserver.key and iamserver.csr.

Step 2: Sign the CSR using your existing Certificate Authority and create a certificate

```
openssl x509 -passin pass:passw0rd -req -days 365 -in iamserver.csr -CA
ca.crt \ -CAkey ca.key -out iamserver.crt -CAcreateserial -CAserial ca.seq
-sha256 -extfile \ <(printf "[SAN]\nsubjectAltName=DNS:iamserver.com")
-extensions SAN
```

pass: is the passphrase you picked in step 1. Please change “DNS:....” to your IAM Server hostname.

This results in the follwing files:
iamserver.crt
iamserver.key

Now, these files go into /iam-service/config/certs, create directory structure if necessary.

Copy your ca*.* files to /iam-service/config/certs/ca, , create directory structure if necessary.

Alternatively, you can use a Let’s Encrypt Certificate:

Configuring the SSL Certificates for the IAM Service Web server using Let’s Encrypt

In this tutorial, we will use publicly trusted Let’s Encrypt Certificates for the Web Server Certificate for the IAM Server. All other Certificates are not accessed directly via a web browser and can therefore be created using a Self Signed CA as described in the Documentation of the AppDevPack. Here, we will do a manual setup for the Let’s Encrypt stuff.

Part 1: Install certbot

Certbot is the recommended tool from let’s Encrypt to be used to manually or automated generation of SSL Certificates using Let’s encrypt. Installing certbot on our linux box involves the following prerequisites:

- Please make sure that your DNS hostname can be resolved externally and certbot can access your VM
- If you have any Web server running on port 80 including Domino, stop the Web server for the certbot operations

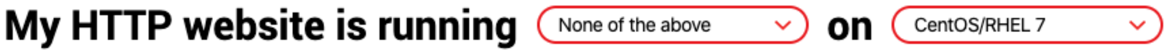
Automating the certbot/Let’s encrypt operations is beyond this tutorial - for testing purposes keep in mind that Let’s encrypt certificates are valid for three (3) months and have to be renewed before expiring. Using certbot manually means the renewal process is straightforward - it’s the same process as the initial setup that needs to be done.

How to install certbot:

Go to this site:

https://certbot.eff.org/instructions

in the line for “My web site is running ” select “



My HTTP website is running None of the above on CentOS/RHEL 7

This will give you the instructions on how to install certbot below. Don’t create a certificate yet !

to create your certificates do.

```
sudo certbot certonly --standalone -d your.dns.name (your web-domain or other
DNS names)
```

with that, Let’s Encrypt will create a bunch of files in /etc/LetsEncrypt/live/<domain or DNS name>

Let's encrypt will generate a set of files:
cert.pem, chain.pem, fullchain.pem, privkey.pem and a README that explains each:

`privkey.pem` : the private key for your certificate.
`fullchain.pem`: the certificate file used in most server software.
`chain.pem` : used for OCSP stapling in Nginx >=1.3.7.
`cert.pem` : will break many server configurations, and should not be used

https://certbot.eff.org/docs/using.html#where-are-my-certificates

After the certificates have been created, make sure to restart your Web server if you had one running on this machine.

Step 2: Configure IAM and PROTON using the Let’s Encrypt Certificates

2.1 IAM Service Certificates for the Web Server

When configuring IAM, the server certificate should be fullchain.pem and the private key file should be privkey.pem. I renamed fullchain.pem to iamserver.crt and privkey.pem to iamserver.key.

Please rename/copy “`fullchain.pem`” to “`iamserver.crt`”
Please rename/copy “`privkey.pem`” to “`iamserver.key`”

Copy `iamserver.crt` and `iamserver.key` to `/home/notes/iam-service/config/certs/`

Change Ownership of files there to notes user if needed !

2.2 Fixing introspection from Proton

Assuming you want proton to be able to invoke the introspection API (for Act-As-User use cases), you need to install a root CA in the proton.kyr file to allow the introspection client code to verify the IAM server certificate. That certificate is here (at the time of this writing):

https://letsencrypt.org/certs/trustid-x3-root.pem.txt.

More details can be found here:

https://letsencrypt.org/certificates/.

Detailed steps for installing root CA:

Start in `/local/notesdata`

1) Getting the root CA:

```
curl https://letsencrypt.org/certs/trustid-x3-root.pem.txt -o le-ca.pem
```

2) Examining the root CA with OpenSSL:

```
openssl x509 -in le-ca.pem -text
```

Yields this output (note the highlighted Subject and Issuer):

```
Certificate
Data:
    Version: 3 (0x2)
Serial Number:
    44:af:b0:80:d6:a3:27:ba:89:30:39:86:2e:f8:40:6b
Signature Algorithm: sha1WithRSAEncryption
    Issuer: O=Digital Signature Trust Co., CN=DST Root CA X3
    Validity:
        Not Before: Sep 30 21:12:19 2000 GMT
        Not After : Sep 30 14:01:15 2021 GMT
    Subject: O=Digital Signature Trust Co., CN=DST Root CA X3
```

3) Importing certificate as a trusted root into the proton keyring file (assuming it's named proton.kyr)

```
cd /local/notesdata
```

```
/opt/ibm/domino/bin/tools/startup /opt/ibm/domino/notes/latest/linux/KYRTOOL
import roots -k proton.kyr -i certs/le-ca.pem
```

Output:

```
Using keyring path 'proton.kyr'
SEC_mpfct_ImportTrustRootToKYR succeeded
```

4) Verify that its really in the keyring:

```
/opt/ibm/domino/bin/tools/startup /opt/ibm/domino/notes/latest/linux/KYRTOOL
show roots -k proton.kyr | grep ".*Digital Signature Trust Co.*"
```

Output:

```
CN=DST Root CA X3/O=Digital Signature Trust Co.
Subject:       CN=DST Root CA X3/O=Digital Signature Trust Co.
Issuer:       CN=DST Root CA X3/O=Digital Signature Trust Co.
```

With this done, we can now go on and set up the IAM Service. For the sake of this tutorial, we will do this on the same VM than our HCL Domino Server. In production, this might be a different machine.

Step 3: Install the IAM Service

3.1 Install Node.js if not yet present

IAM's strategy is to support the LTS version of Node.js. The currently supported version is V10.x, which can be found from the official site.

3.2 Hostname

After creating the SSL certificates in the previous chapter, you should know that one by now ;-)

3.2.1 Setup Domino Credential Store

Besides the ID-Vault and Transactional Logging, the IAM Service also utilizes the Domino Credential store application to store secure ids and secrets to do introspections of access tokens from the proton task or the DSAPI-Filters for the Domino REST-Services. To do this, a credential store must be configured on our Domino Server that acts as a resource provider to the IAM Service. The following procedure guides you through that setup process in the Domino Admin Client.

Note: You can omit this step if you already have a credential store configured on your Domino server.

- At the Domino server console, use the `keymgmt create nek` command to create the document encryption key in the Domino server ID file. Sample:

  To create a document encryption key called `credstorekey`, to be used to secure a credential store, enter:

```
KEYMGMT CREATE nek credstorekey
```

- Check the server console log and make sure you see the following message:

  `NEK credstorekey created successfully`

  Make note of the displayed fingerprint for the key.
- Use the `keymgmt create credstore` command to create the credential store application and assign the document encryption key.

  To create the credential store using a document encryption key called `credstorekey`, enter:

  ```
  KEYMGMT CREATE credstore credstorekey
  ```
- Make sure the displayed fingerprint matches the one you made note of in the previous step.
- Make sure the Domino server \data directory now has a directory \IBM_CredStore.
- Make sure credstore.nsf exists in the directory.

3.3 Configure Domino to host the IAM database

3.3.1 Proton configuration - this should have been done in the previous chapters

Make sure you are using SSL for Proton and Client certificate authentication

```
PROTON_SSL=1
PROTON_AUTHENTICATION=client_cert
```

should be present in the Notes.ini

3.3.2 Setup ID-Vault if not already done

- As a Domino Admin, start the Admin Client and switch to the Administration Server of your Domino Directory.
- From the Admin Client Click **Tools > ID Vaults > Create**
- Give the Vault a Name
- Tell the process where to store the Vault ID file and set up a password for it
- Select the primary Domino Server for the ID Vault
- Define at least one vault administrator
- Add organizations that trust the vault for ID storage -> you need the respective certifier ids for this step. This creates a policy that gets added to new users (we need that in the steps ahead for the technical users for proton and IAM)

More details can be found here:

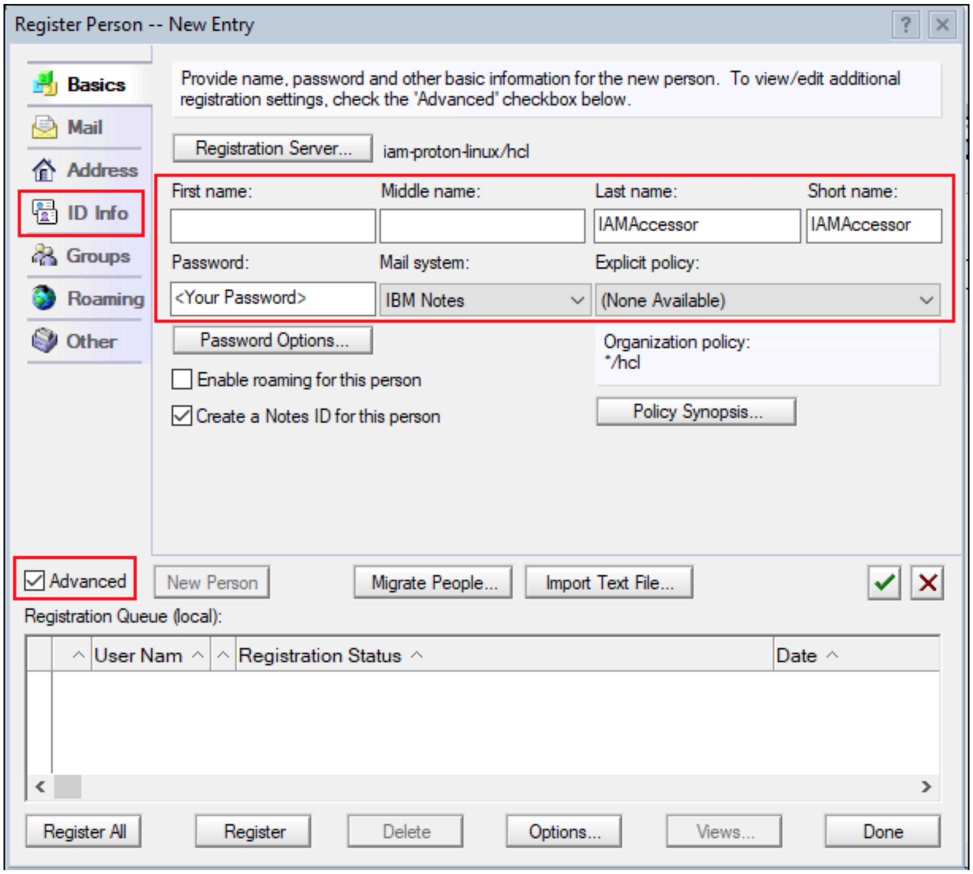
https://www.ibm.com/support/knowledgecenter/en/SSKTMJ_9.0.1/admin/conf_notesidvault_c.html

3.3.3 SSL Certificates for the server

We did prepare those in Step 1 and 2 of this tutorial. Make notes where to find them when needed. You should have them in `/home/notes/iam-service/config/certs`

3.3.4 Functional Notes ID for IAM

Make sure to do this AFTER setting up ID-Vault. The technical user for IAM has to be in the ID-Vault or the introspection sequence from Domino to IAM will not work.



Register Person -- New Entry

Provide name, password and other basic information for the new person. To view/edit additional registration settings, check the 'Advanced' checkbox below.

Registration Server... iam-proton-linux/hcl

First name:
Middle name:
Last name: IAMAccessor
Short name: IAMAccessor
Password: <Your Password>
Mail system: IBM Notes
Explicit policy: (None Available)

Organization policy: */hcl

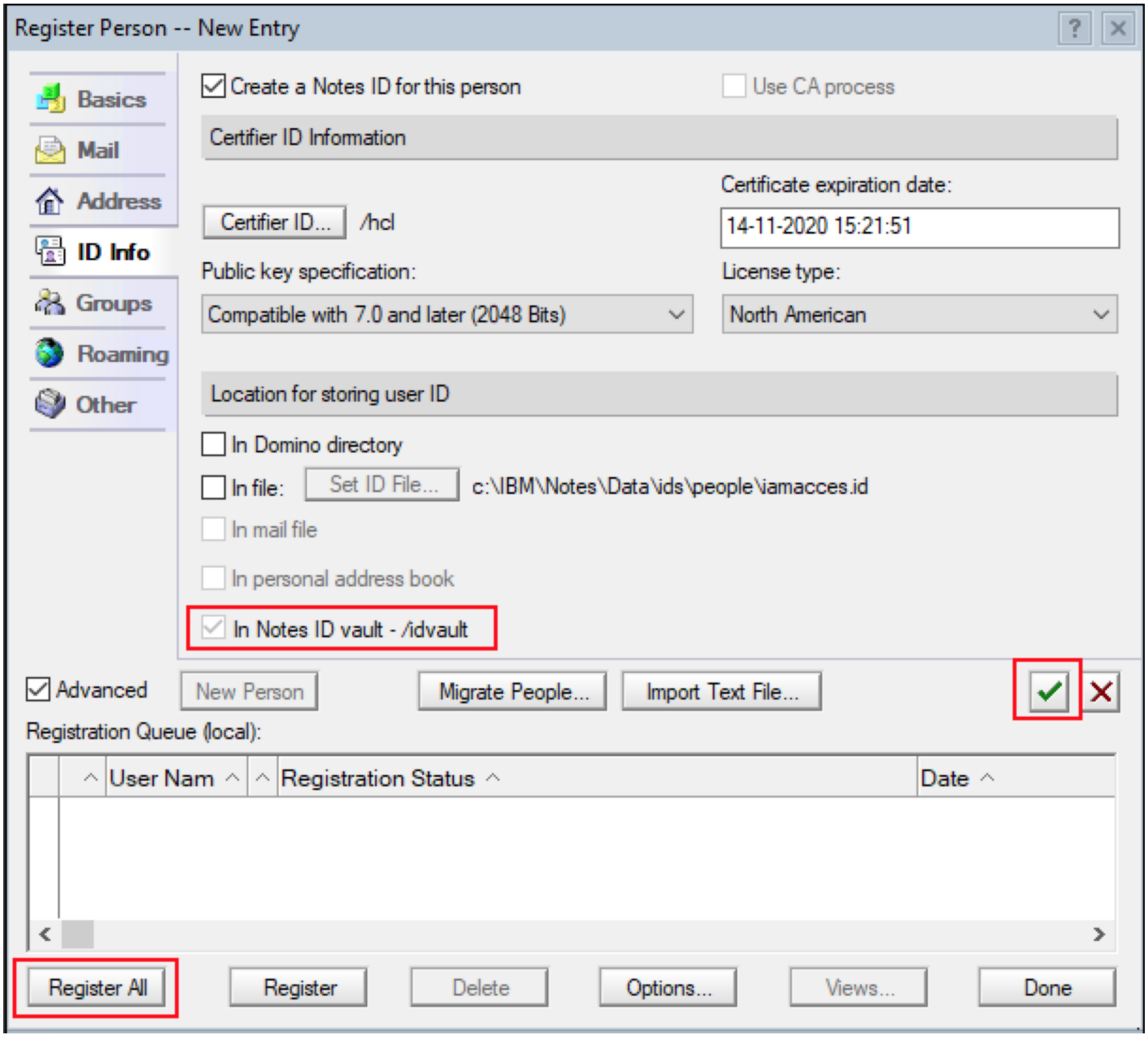
□ Enable roaming for this person
☑ Create a Notes ID for this person

☑ Advanced

Registration Queue (local):

User Nam	Registration Status	Date

Make sure that the ID-Vault is selected:



Register Person -- New Entry

☑ Create a Notes ID for this person

□ Use CA process

Certifier ID Information

Certifier ID... /hcl

Certificate expiration date: 14-11-2020 15:21:51

Public key specification: Compatible with 7.0 and later (2048 Bits)

License type: North American

Location for storing user ID

□ In Domino directory

□ In file: Set ID File... c:\IBM\Notes\Data\ids\people\iamacces.id

□ In mail file

□ In personal address book

☑ In Notes ID vault - /idvault

☑ Advanced

Registration Queue (local):

User Nam	Registration Status	Date

For the next step, we need to create an SSL Certificate using OpenSSL. In case you do not have that installed yet, please check here and follow the instructions for setup:

https://webhostinggeeks.com/howto/how-to-install-and-update-openssl-on-centos-6-centos-7/

We start again in `/local/notesdata` as this is where your proton self-signed CA files should be stored (`ca.key, ca.crt, ca.seq`). Please look them up before starting this next part.

We have to create an Internet Certificate for this user to be imported into the Domino Directory for domino-db to be able to use the technical user id from IAM. This certificate is again internal only - it will never hit a browser so we can use the same self-signed-CA that we used to sign the other proton certificates before. Here's the sequence to create the certificate:

```
openssl genrsa -des3 -out iamuser.key 4096
```

Set a pass phrase for the key-file.

In the following line, please replace “common name” with the canonical name of the technical user like so:

```
/o=hcl/CN=IAMUser
```

Then, we create a certification request (CSR):

```
openssl req -new -key iamuser.key -out iamuser.csr -subj "common name"
-sha256
```

Then,we sign the certificate with the PROTON CA:

Depending on your OpenSSL config, you might have to do this step using the root user. Please also make sure you have the pass phrase for you ca.key available.
Hint: They might be in the `make_certs.sh` file from creation of the CA. This file is located in `/opt/ibm/domino/notes/latest/linux`

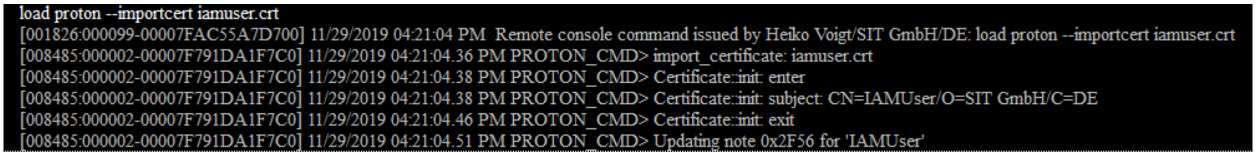
```
openssl x509 -req -days 365 -in iamuser.csr -CA ca.crt -CAkey ca.key -out iamuser.crt -CAserial ca.seq
```

The last step here is to import the client certificate into the person document of the technical user:

This is done on the domino server console:

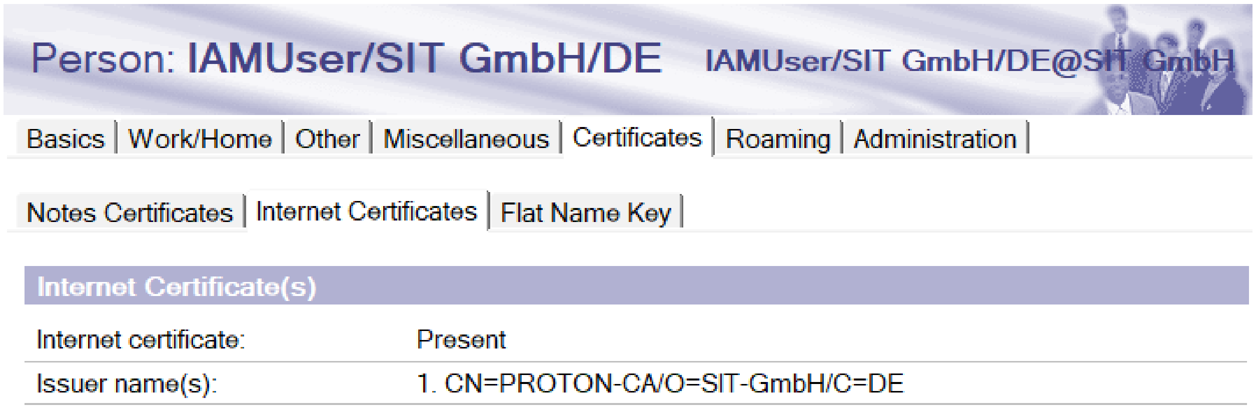
```
load proton --importcert iamapp.crt
```

On the console, you should see something similar to this:



```
load proton --importcert iamuser.crt
[001826:000099-00007FAC55A7D700] 11/29/2019 04:21:04 PM  Remote console command issued by Heiko Voigt/SIT GmbH/DE: load proton --importcert iamuser.crt
[008485:000002-00007F791DA1F7C0] 11/29/2019 04:21:04.36 PM PROTON_CMD> import_certificate: iamuser.crt
[008485:000002-00007F791DA1F7C0] 11/29/2019 04:21:04.38 PM PROTON_CMD> Certificate::init: enter
[008485:000002-00007F791DA1F7C0] 11/29/2019 04:21:04.38 PM PROTON_CMD> Certificate::init: subject: CN=IAMUser/O=SIT GmbH/C=DE
[008485:000002-00007F791DA1F7C0] 11/29/2019 04:21:04.46 PM PROTON_CMD> Certificate::init: exit
[008485:000002-00007F791DA1F7C0] 11/29/2019 04:21:04.51 PM PROTON_CMD> Updating note 0x2F56 for 'IAMUser'
```

Please check in the person document on the tab internet certificates whether the certificate got installed correctly.



Person: IAMUser/SIT GmbH/DE IAMUser/SIT GmbH/DE@SIT GmbH

Internet Certificate(s)

Internet certificate:	Present
Issuer name(s):	1. CN=PROTON-CA/O=SIT-GmbH/C=DE

3.3.5 Create the IAM storage database

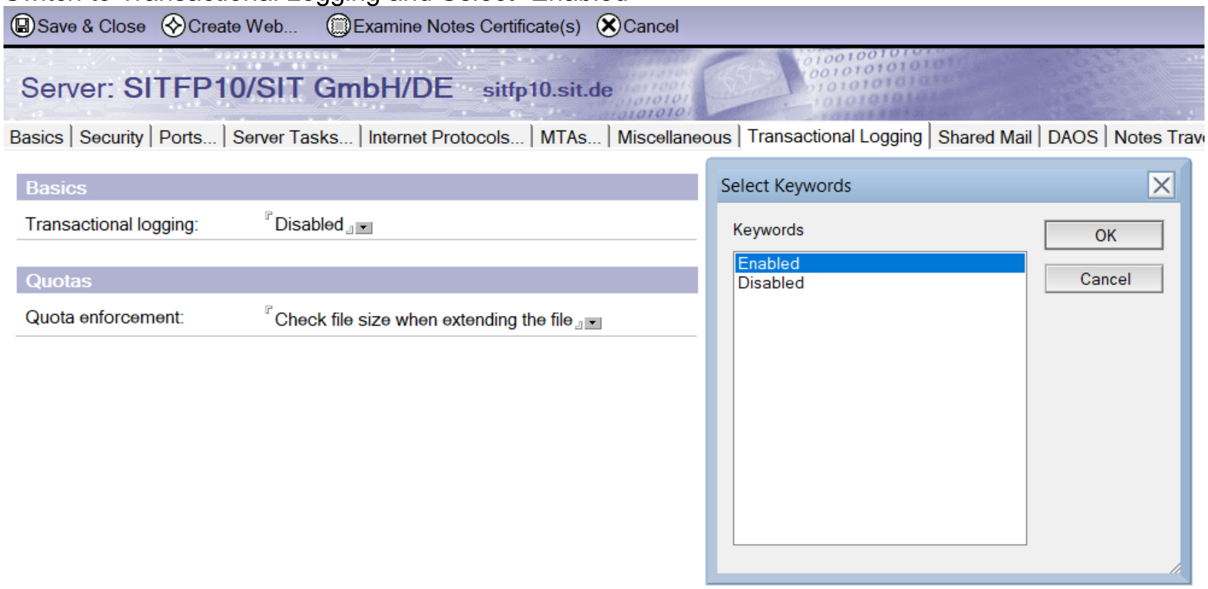
3.3.5.1 Transaction logging

Make sure to enable transaction logging on your PROTON domino server

-> Link/Write Up on how to do that

https://www.ibm.com/support/knowledgecenter/en/SSKTMJ_10.0.1/admin/admn_settingupadominoserverfortransactionlogging_t.html

- Open your current server document
- Switch to Transactional Logging and Select “Enabled”



Server: SITFP10/SIT GmbH/DE sitfp10.sit.de

Basics

Transactional logging: Disabled

Quotas

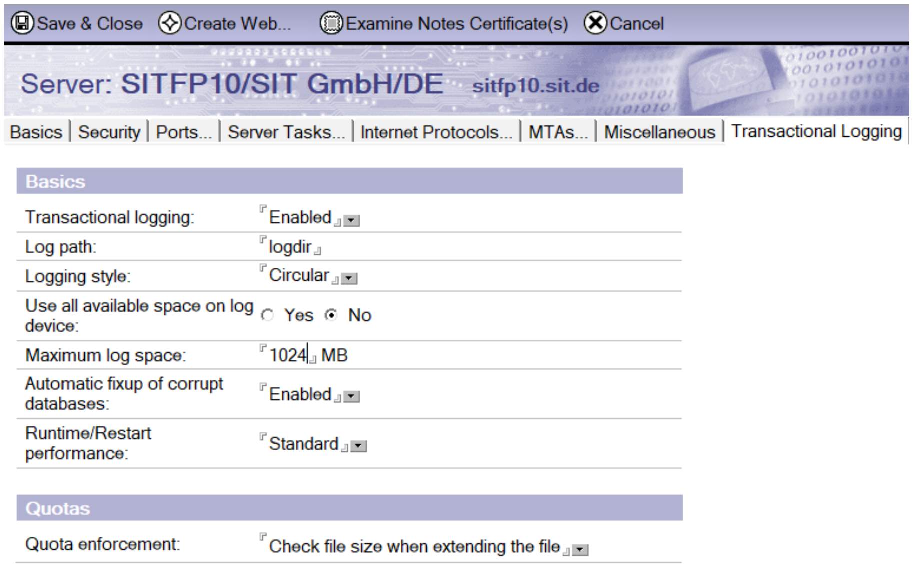
Quota enforcement: Check file size when extending the file

Select Keywords

Keywords

Enabled
Disabled

- In the following settings page, leave everything as-is for our testing environment, the only change I made was for max log size:


Save & Close
Create Web...
Examine Notes Certificate(s)
Cancel
Basics | Security | Ports... | Server Tasks... | Internet Protocols... | MTAs... | Miscellaneous | Transactional Logging

Server: SITFP10/SIT GmbH/DE sitfp10.sit.de

Basics

Transactional logging:	Enabled
Log path:	logdir
Logging style:	Circular
Use all available space on log device:	○ Yes ● No
Maximum log space:	1024 MB
Automatic fixup of corrupt databases:	Enabled
Runtime/Restart performance:	Standard

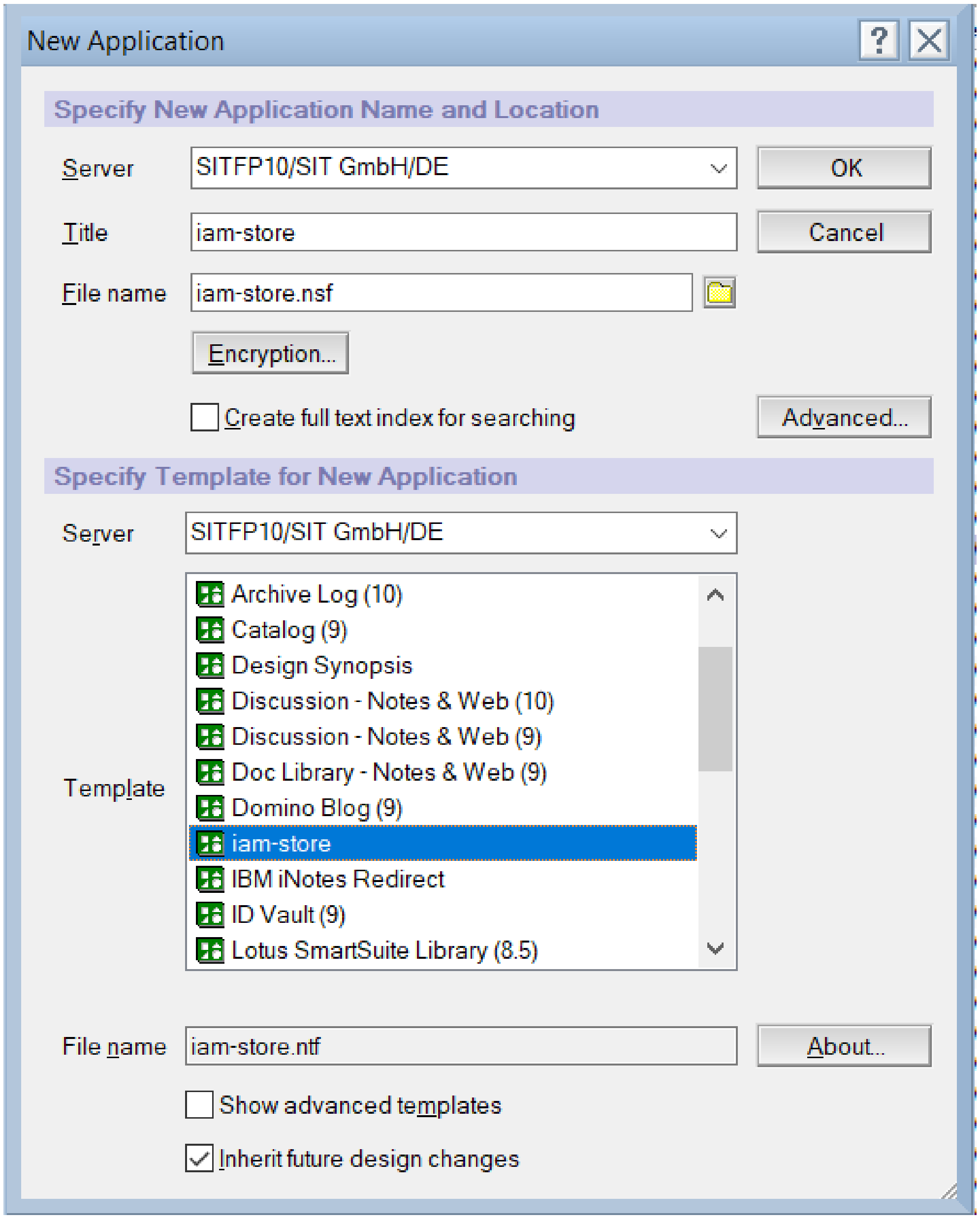
Quotas

Quota enforcement:	Check file size when extending the file

- Klick “Save & Close”
- Restart your Domino Server to get the changes in effect.

3.3.5.2 Create the IAM storage database

1. Copy the iam-store.ntf into Domino server /local/notesdata directory.
2. Fix the ACL as needed
3. Sign the template with an Admin ID.
4. Open the Files tab in the IBM Domino Administrator.
5. Click Top Left menu: File -> Application -> New...
6. In the opened dialog, choose your server name from the drop-down list, input the title as iam-store, and update the File name to iam-store.nsf.
7. In the 'Specify Template for New Application' section, choose your server and select the 'iam-store.ntf' template.
8. Click OK to create the application.
9. In Files tab, press F9 to refresh the file list to see iam-store.nsf.
10. Fix the ACL of the Database as needed.



New Application

Specify New Application Name and Location

Server: SITFP10/SIT GmbH/DE

Title: iam-store

File name: iam-store.nsf

Encryption...

□ Create full text index for searching

Specify Template for New Application

Server: SITFP10/SIT GmbH/DE

Template:
- Archive Log (10)
- Catalog (9)
- Design Synopsis
- Discussion - Notes & Web (10)
- Discussion - Notes & Web (9)
- Doc Library - Notes & Web (9)
- Domino Blog (9)
- iam-store
- IBM iNotes Redirect
- ID Vault (9)
- Lotus SmartSuite Library (8.5)

File name: iam-store.ntf

□ Show advanced templates

☑ Inherit future design changes

3.3.5.3 **Give the IAM's functional ID access to the database**

IAM's functional ID needs to be added to the DB's ACL:

1. In Domino Administrator, in Files tab, find and select the iam-store.nsf.
2. Click the Manage ACL... from the right side bar under Database section.
3. Click 'Add...' button to add the Functional ID in the opened dialog
   (in our sample: IAMUser/SIT GmbH/DE).
4. Check the [_ReadAllItems] role for the above Function ID you added and click OK.
5. Default Access is Designer - Editor should be sufficient.

3.3.5.4 Add the IAM Storage Database to the design catalog

On your domino server console, issue the following command:

```
load updall iam-store.nsf -e
```

3.3.6 IAM Configuration

Decompress the package domino-iam-service.tgz package to a destination, where you will be running it from later on. My example is

```
/home/notes/iam-service
```

in case you will be using the Notes User to run IAM as well.

in preparation for SSL communication copy the following certificates into the folder

```
../iam-service/config/certs
```

1. copy iamserver.key and iamserver.crt from your Let’s Encrypt folder.
2. copy IAMUser.key and IAMUser.crt into ../certs
3. Create the “ca” folder if it doesn’t exists in `../iam-service/config/certs`
4. copy the PROTON CA (ca.crt, ca.key) into `../iam-service/config/certs/ca`

Holy mackerel, now we can finally start to configure the IAM Service.

In the `../iam-service` directory, start with the following command:

a couple of packages for Node.js might be missing so we start of by doing:

```
npm install --save
```

and

```
npm audit fix
```

To fix vulnarabilities

Then, we start with:

```
npm run setup
```

Choose your installation type:

```
[16:52:48][info][configStore][master]: No configurations found in configuration directory
Welcome to IAM Setup
? Choose setup mode (Use arrow keys)
  Production.
  Pilot, a helper mode to pilot IAM service.
```

use “Production”.

First thing when the setup starts is to create an admin user and a password for the IAM Service.

```
Welcome to IAM Setup
? IAM Admin username: admin
? Enter IAM Admin password: [hidden]
? Enter IAM Admin password again: [hidden]
Admin password is set successfully
```

That’s the output you should see.

The next step is to create a password for the IAM Server for Startup:

```
? Enter IAM server password: [hidden]
? Enter IAM server password again: [hidden]
Server password is set successfully
```

Setting Name	Setting Description
IAM Service ISSUER	Either the server URL (if local server) or a proxy URL (if load balancer is used).

```
------------------------
About to config The URL of the issuer.
? IAM Service ISSUER: https://sitfp10.sit.de/

Following configuration have been input for The URL of the issuer --

issuer - https://sitfp10.sit.de:9443
```

The next part defines the URL for the IAM Service.

A word on Fail over and Load Balancing for the IAM Service. You can set up multiple IAM Service instances. To use them for Load Balancing and Fail over operations, you would place a load balancer (web server, sprayer) in front of them. In this case, the Endpoints for the IAM Service have to be on the load balancer not the specific endpoints of each instance ! This is a pretty common pattern for application servers other than domino.

In this section we will also configure the ports on where the IAM Service will be available. The default ports are 443 and 8443 - please check if you already have a Web server running on 443 and if so, change the port for IAM to another one. In my example, I will use port 9443. Keep in mind that you have to open this port on your VMs firewall to make it accessible from outside.

Setting Name	Setting Description
IAM service's host	The host that IAM service will listen on. '[default]': accept the request from all network interfaces; '0.0.0.0': accept all requests from IPv4; hostname/ip: only accept the request from a specified host.
IAM service's port	The IAM service port number
ADMIN service's host	The host that IAM Admin service will listen on. '[default]': accept the request from all network interfaces; '0.0.0.0': accept all requests from IPv4; hostname/ip: only accept the request a specified host.
ADMIN service's port	The IAM Admin service port number

The following dialogue is used to configure the ports and URLs:

```
About to config Address Settings.
? IAM service's host(leave as '[default]' to accept the request from all
network interfaces):
? IAM service's port: 9443
? ADMIN service's host(leave as '[default]' to accept the request from all
network interfaces):
? ADMIN service's port: 8443

Following configuration have been input for Address Settings --

IAM service's host - [default]
IAM service's port - 9443
ADMIN service's host - [default]
ADMIN service's port - 8443

? Confirm? Yes
```

The next step is the configuration of the SSL Settings.

Setting Name	Setting Description
IAM server's SSL key file path (relative to 'config/certs' folder)	IAM server's private key -> our Let’s Encrypt Key: iamserver.key
Server's SSL key password	the key file does not have a password
Server's SSL cert file path (relative to 'config/certs' folder)	our Let’s Encrypt certificate: iamserver.crt

Use the following dialogue to add this information:

```
About to config SSL Settings.
? IAM server's SSL key file path (relative to 'config/certs' folder): con-
```

```
fig/certs/iamserver.key
? Server's SSL key password: <leave blank>
? Server's SSL cert file path (relative to 'config/certs' folder): config/certs/iamserver.crt

Following configuration have been input for SSL Settings --

IAM server's SSL key file path - config/certs/iamserver.key
Server's SSL key password -
Server's SSL cert file path - config/certs/iamserver.crt

? Confirm? Yes
```

Next up is the configuration for the domino data store

Setting Name	Setting Description
Hostname of the Domino server	The Domino server's hostname
Domino's Proton service port	The proton connection port. The same port that is configured on Domino server
IAM's storage NSF file path (e.g.: <iam-store.nsf>)	The database name including the path. The path is relative to the Domino data directory.
IAM's Proton client cert file path (relative to 'config/certs' folder)	client certificate
IAM's Proton client cert key file path (relative to 'config/certs' folder)	client private key
The key file's protection pass phrase	client private key password
The cert's bounded functional ID's Notes password	ID file password

```
About to config Storage Settings.

? Hostname of the Domino server: domino-server.hcl
? Domino's Proton service port: 3002 (in our sample)
? IAM's storage nsf file path, a relative path to Domino data path: iam-store.nsf
? IAM's Proton client cert key file path (relative to 'config/certs' folder): config/certs/iamaccessor.key (in our sample)
? The key file's protection passphrase: [hidden]
? IAM's Proton client cert file path (relative to 'config/certs' folder): config/certs/iamaccessor.crt (in our sample)
? The cert's bounded functional ID's Notes password: ***

Following configuration have been input for Storage Settings --

Domino server hostname - domino-server.hcl
Domino Proton service port - 447
Storage nsf file - iam-store.nsf
key for the client cert - config/certs/proton.key
```

```
The key file's protection passphrase: - ***
Domino Proton client cert - config/certs/proton.crt
Functional ID's Notes password - ***

? Confirm? Yes
```

That concludes the basic setup for IAM and you will you will be asked to confirm your configuration:

```
? Do you want to change some configuration? No, the listed above will be saved
to local.properties. IAM Server setup completed.
NOTICE: The configuration file has been encrypted.
NOTICE: Original configuration file has been renamed to *.[timestamp].bak
file, please delete it after confirmation to keep confidential.
```

3.4 First Time Startup

When these steps are done, please start the IAM Service go to your console, cd into ../iam-service and start the IAM Service using

```
npm start
```

You will be prompted for the IAM Server password that you configured during setup:

```
Start to unlock config:
? Enter current IAM server password: [hidden]
```

Then you should see similar messages that tell you that the server started successfully:

```
[01:48:05][info][initServices]: Start IAM service on allAddress:9443
[01:48:05][info][initServices]: IAM service serves on port 9443
[01:48:05][info][initServices]: Start Admin service on allAddress:8443
[01:48:05][info][initServices]: Admin service serves on port 8443
```

You can now visit the URL Endpoints for your server:

The IAM-Server is available on:

```
https:://<your.iam-server.domain>:9443
```

The Admin Interface is available on:

```
https:://<your.iam-server.domain>:8443
```

3.5 Configure IDP to connect with LDAP

To configure an Identity Provider (LDAP), log into the IAM admin interface using the IAM Admin user id and credentials that you configured during setup. In our case, the Admin interface is accessible at

https://<your-iam-server-dns-hostname>:8443

You have the choice to configure Domino or Active Directory as LDAP Server. In our sample, we use Domino.

To configure the IDP, click `configuration` in the left panel and click the `Configure LDAP` button in the Dashboard.

To configure the LDAP Server, you will need the following information:

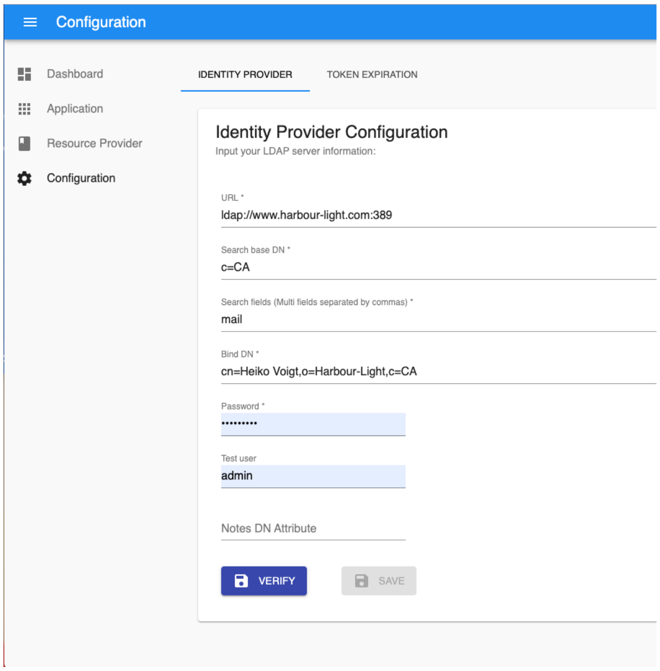
Field	**Description**	**Example**
URL	The URL of the LDAP server, including the full qualified DNS name and the port. You can skip the port part if LDAP server uses the default port(389 for ldap, 636 for ldaps)	ldap://our.domino-server.com:389

Base DN	The base DN from which to search for the provided user credentials	o=myorg*
		*Attention: if your Canonical names in Domino contain a country code, your Base DN is the country code e.g.:
		c=de or c=ca

Search Fields	The field to search for the provided user credentials	mail
Bind DN	The DN to bind to for performing ldapsearch	cn=ldapAdmin,ou=company,o=com,c=CA
Password	The password for the LDAP Bind DN	
Test user	Used for LDAP verification. Input an existing user in LDAP directory	testuser@myorg.com
Notes DN Attribute	Optional. Only used when using Active Directory as IDP,and Directory assistance is configured on Domino to look up user information in Active Directory.	description

You can test the LDAP Access with be adding a test user and click “verify”. After saving your settings, please restart the IAM Server for the settings to take effect.

Here’s a sample Screen:



Identity Provider Configuration

Input your LDAP server information:

URL *
ldap://www.harbour-light.com:389

Search base DN *
c=CA

Search fields (Multi fields separated by commas) *
mail

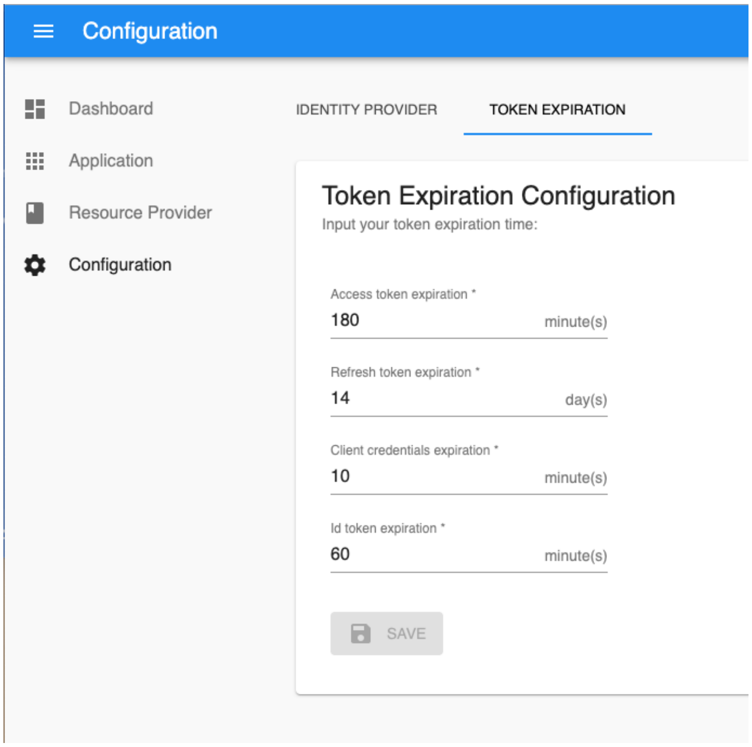
Bind DN *
cn=Heiko Voigt,o=Harbour-Light,c=CA

Password *
•••••••••

Test user
admin

Notes DN Attribute

In the 2nd Configuration page you can alter the Token Expiration settings for Access and Refresh Tokens etc. I left this to the defaults which are named in the documentation of the AppDev Pack:



Token Expiration Configuration

Input your token expiration time:

Access token expiration *
180 minute(s)

Refresh token expiration *
14 day(s)

Client credentials expiration *
10 minute(s)

Id token expiration *
60 minute(s)

Attention: if you make chances to these settings and save them, make sure to restart the IAM Service for the changes to take effect.

3.6 Domino Resource Provider

A Domino resource provider receives an access token from an application and uses an IAM introspection request to acquire information about the access token. Before it can make an introspection request, the resource provider must first be registered with the IAM service. During the registration process, IAM creates a resource provider id and secret to be used in each introspection request.
There are currently two kinds of Domino resource providers:

- Proton is a resource provider when handling a request in which the client application is acting as a user.
- The OAuth DSAPI extension is a resource provider when handling a Domino Access Services (DAS) request that includes an access token.

Every IAM service instance should have at least one Domino resource provider configured, but you don't necessarily need to configure both types of resource providers. For example, if your IAM service instance will never be used to introspect an access token for a DAS request, there is no need to configure the OAuth DSAPI extension.

SUMMARY: IF YOU WANT TO USE THE ACT-AS-USER FEATURE or the DSAPI Filters you HAVE to configure at least one domino resource provider !

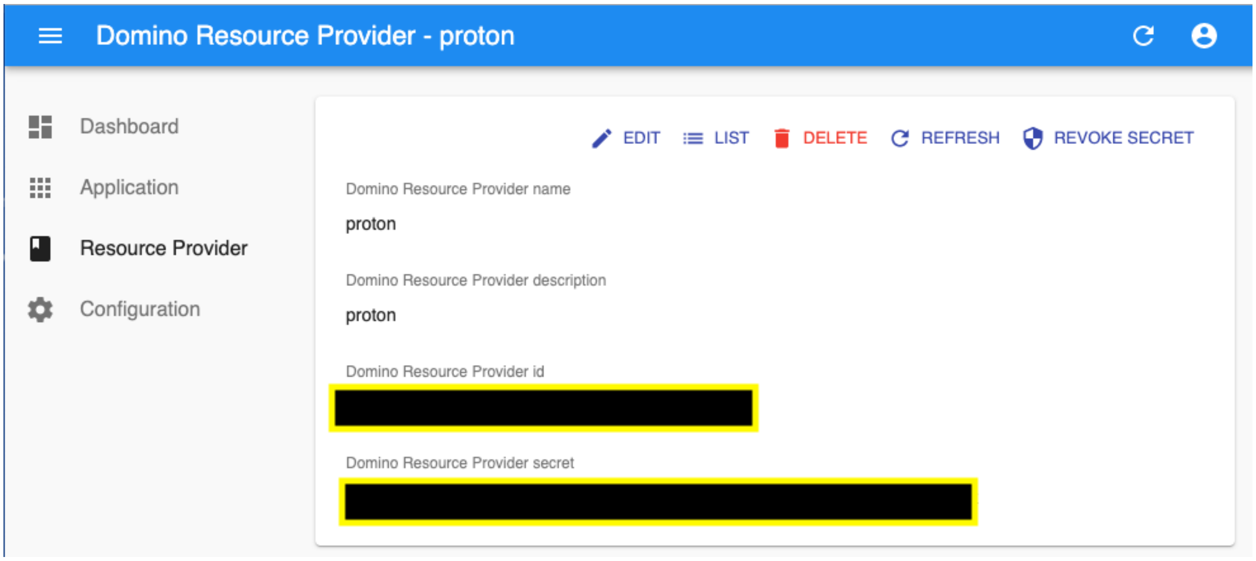
Before configuring applications, we will take a look on how to complete the IAM Configuration by setting up our Domino Server as a Resource Provider to allow the usage of Domino REST-APIs with IAM and also the introspection of user ids between IAM and Domino for the act-as-user feature.

To create a Resource Provider select “Resource Provider” from the Configuration menu (see above). The click the CREATE button to register a new resource provider.

These are the fields you need to fill in:

Field	**Description**
Domino Resource Provider name	Name of Domino resource provider
Domino Resource Provider description	A short description of the Domino resource provider

Here is a sample of my settings:



Domino Resource Provider name

proton

Domino Resource Provider description

proton

Domino Resource Provider id

[redacted]

Domino Resource Provider secret

[redacted]

Please note !

The fields `Domino Resource Provider Id` and Domino Resource Provider secret will be auto-calculated. We will need those during the setup for the OAuth DSAPI Extension for the Domino Web server and the Act-as-User-Feature.

To use these values, we have to add the credentials to the Domino Credential Store created previously in step 3.2.1.

To import the Resource Provider ID into the credential store, we have to go to the command line:

cd /local/notesdata

Then we issue the following command (this is the resource provider for the proton task for the act as user feature):

```
/opt/ibm/domino/bin/tools/startup /opt/ibm/domino/notes/latest/linux/oauthcfg
create proton <resource provider id> <resource provider secret> https://<your-
iam-server>:9443/token/introspection
```

For the DSAPI Filters we repeat that command using DEFAULT as the name:

```
/opt/ibm/domino/bin/tools/startup /opt/ibm/domino/notes/latest/linux/oauthcfg
create DEFAULT <resource provider id> <resource provider secret>
https://<your-iam-server>:9443/token/introspection
```

To check if everything went ok, we will list the Resource provider credentials by using this command:

```
cd /local/notesdata
```

```
/opt/ibm/domino/bin/tools/startup /opt/ibm/domino/notes/latest/linux/oauthcfg
```

```
list proton
```

```
/opt/ibm/domino/bin/tools/startup /opt/ibm/domino/notes/latest/linux/oauthcfg list DEFAULT
```

The documentation of the AppDevPack tells you how to remove the resource provider definitions.

3.7 DSAPI-Filter setup

Finally, the last piece of the puzzle is to install the DSAPI-Filters IN CASE YOU NEED THEM. They are not necessary for domino-db (Node.js) based access to Domino, they are needed if you want to access Domino REST-APIs like Free/Busytime and Domino Access Services via the Domino Web server

To do this, we have to make sure we have the file

`“liboauth-dsapi.so”`

in `/opt/ibm/domino/notes/latest/linux`

The current documentation of the AppDevPack is wrong here - there is no separate file to extract, the file should be there as soon as you extract the proton add in into the directory noted above. Please make sure it is there.

Next, we have to add a Keyring File in PEM format to the data directory that contains the necessary Certificates to allow Domino to verify the IAM Service Certificates. This can be our `iamserver.crt` from Step 2. Copy the file into `/local/notesdata` if you not have done so earlier.

So we add the following to the Notes.ini:

```
OAUTH2_DSAPI_KEYRING=iamserver.crt
```

Having done that, there are only a couple of steps left:

- Open the Domino directory for the server and locate the server document.
-
- Select the "Internet Protocols" tab and click "Edit Server".
-
- On the http tab, add oauth-dsapi to "DSAPI filter file names"
-
- Save the server document.
-
- Enable the Calendar and Freebusy rest API's using steps found here: Enabling Domino Access services.
-
- On the Domino server console type `restart task http`.

  You should see:
  ```
  HTTP Server: DSAPI OAuth DSAPI Filter version <version> Loaded successfully
  ```

3.8 Famous last words on the Act-As-User Feature

Following this guide, we set up everything you need to use the Act-As-User Feature of the IAM Service. To make use of it, there are some small steps to be done:

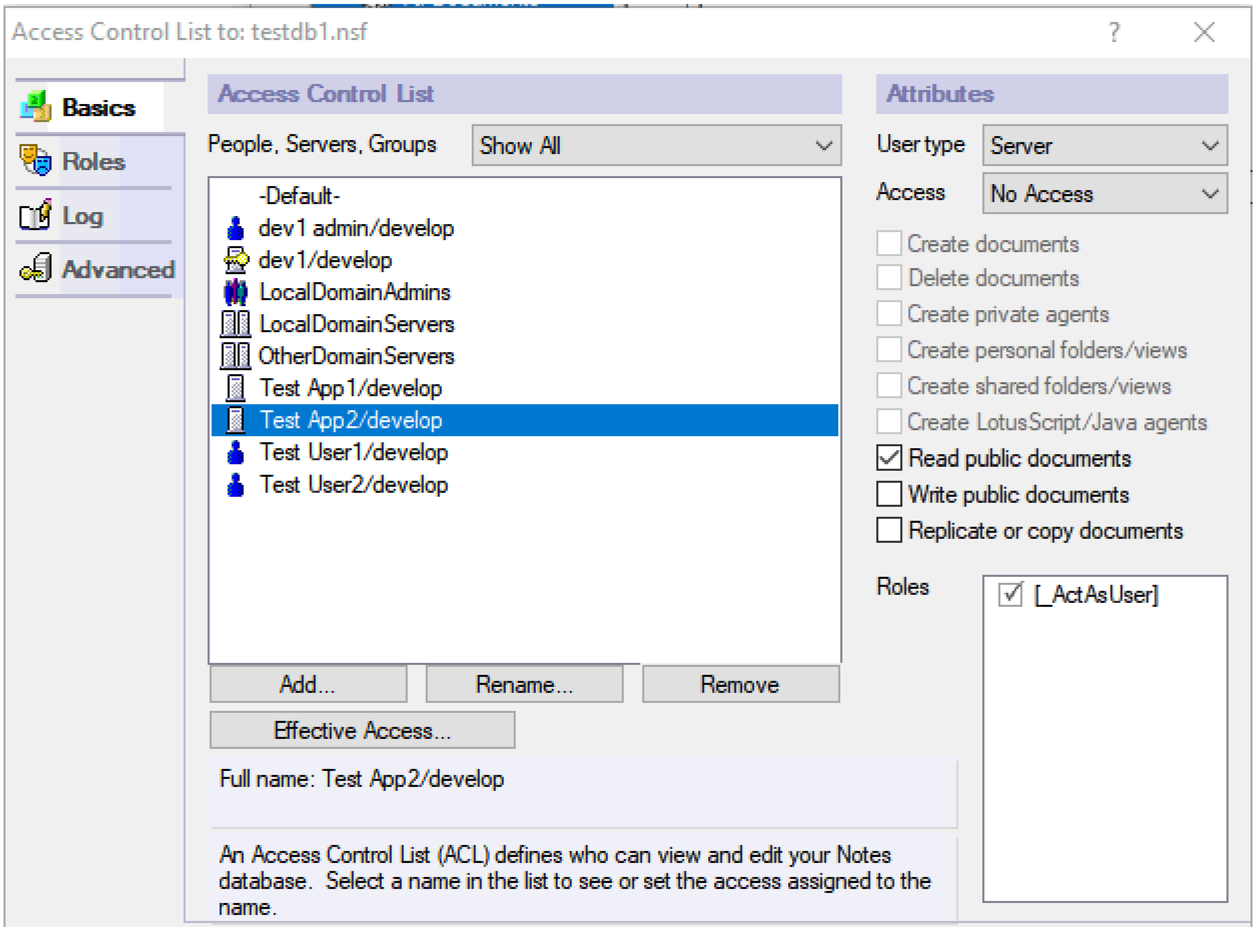
Add the following lines to the Notes.ini:

```
PROTON_ACTASUSER=1
PROTON_IAMCLIENT_CONFIG_NAME=proton
```

This closes the loop to our Resource Provider configuration we did earlier.

Furthermore, to use that feature inside of your Notes Applications, you have to add a role to each database and to add a technical user* for IAM to ACL and add the role to this user like so:

The user role is [_ActAsUser]



Access Control List to: testdb1.nsf

People, Servers, Groups: Show All

- -Default-
- dev1 admin/develop
- dev1/develop
- LocalDomainAdmins
- LocalDomainServers
- OtherDomainServers
- Test App1/develop
- Test App2/develop
- Test User1/develop
- Test User2/develop

User type: Server

Access: No Access

☐ Create documents
☐ Delete documents
☐ Create private agents
☐ Create personal folders/views
☐ Create shared folders/views
☐ Create LotusScript/Java agents
☑ Read public documents
☐ Write public documents
☐ Replicate or copy documents

Roles: ☑ [_ActAsUser]

Full name: Test App2/develop

An Access Control List (ACL) defines who can view and edit your Notes database. Select a name in the list to see or set the access assigned to the name.

The minimum access that that an OAUTH Application needs is "`Read Public Documents`" as in the sample above.

*Make sure that the technical user has been created using the ID Vault !

That’s it !!!!

Addendum: Some Automation

To do some automation, starting the IAM Service in a SSH Window is not ideal. At least I came to a some automation parts that make life easier.

First of all, we create a file password.txt in `/home/notes/iam-service/` using the notes user using:

```
nano password.txt
```

type the IAM Service server password into the file and save it.

Now, we have to export an Environment variable to point the IAM-Service to read the file on Startup. To do this, we switch to the root user and ssh into the VM. On the console issue the following commands:

```
echo export IAM_SERVER_PASSWORD_FILE=/home/notes/iam-service/password.txt>/etc/profile.d/iamserver.sh
chmod 0755 /etc/profile.d/iamserver.sh
```

This will export the environment variable and will keep it alive after restarting the VM.

Now, we can switch back to the Notes User and start the IAM Service as a background task like so:

```
cd /home/notes/iam-service
node iam-server.js &
```

The server should now pick up our password from the file and start as planned. You should see something like this on the console when starting the IAM Service:

```
Start to unlock config:
Env 'IAM_SERVER_PASSWORD_FILE' detected, try to load password from file:
/home/notes/iam-service/password.txt
Config is unlocked.
```

Part 4: A sample application to test OAUTH and Act-As-User Feature

The next step is to come up with a sample application to test the interaction between domino-db and OAUTH2 especially when using the Act-As-User-Feature. If that’s working - you made it.

For that purpose I extended the sample authorization_code_flow example that is part of the AppDevPack. Please check here to find the basic source code:

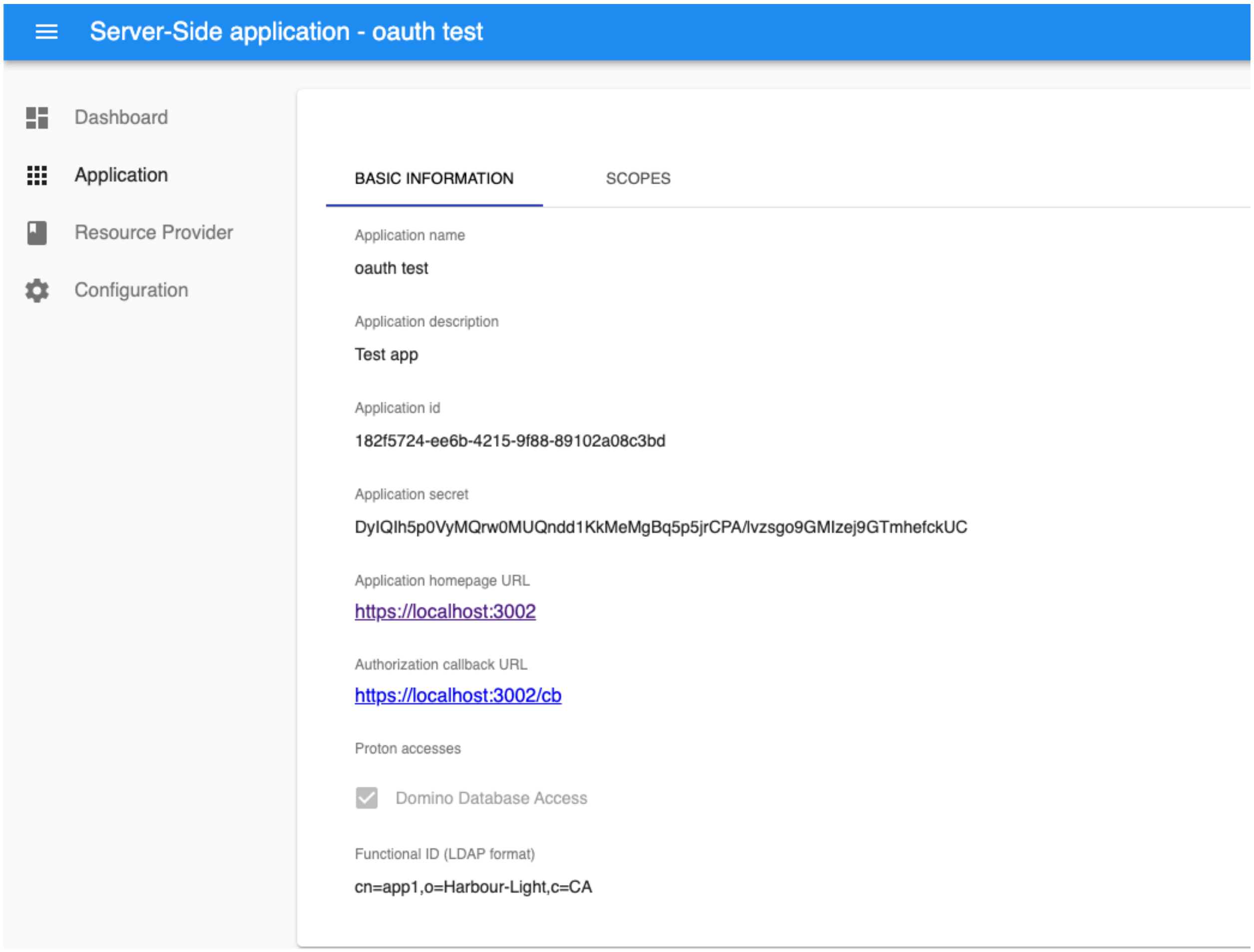
https://doc.cwpcollaboration.com/appdevpack/docs/en/iam_client_library_examples.html

We will be using the Authorization Code Flow example. To start, we will create a IAM Application definition and will use the parameters of this application in the sample code. To do so, log into the IAM Service Administration page and select “Application” from the menu on the left. Then select “Create”. Please fill in a name and a description. Then select the home page url which will be
https://localhost:3002

your Application callback URL will be:

https://localhost:3002/cb

Please select “Domino Database Access” and enter a functional ID in LDAP Format. The final thing should look similar to this:



Application name

oauth test

Application description

Test app

Application id

182f5724-ee6b-4215-9f88-89102a08c3bd

Application secret

DyIQIh5p0VyMQrw0MUQndd1KkMeMgBq5p5jrCPA/lvzsgo9GMIzej9GTmhefckUC

Application homepage URL

https://localhost:3002

Authorization callback URL

https://localhost:3002/cb

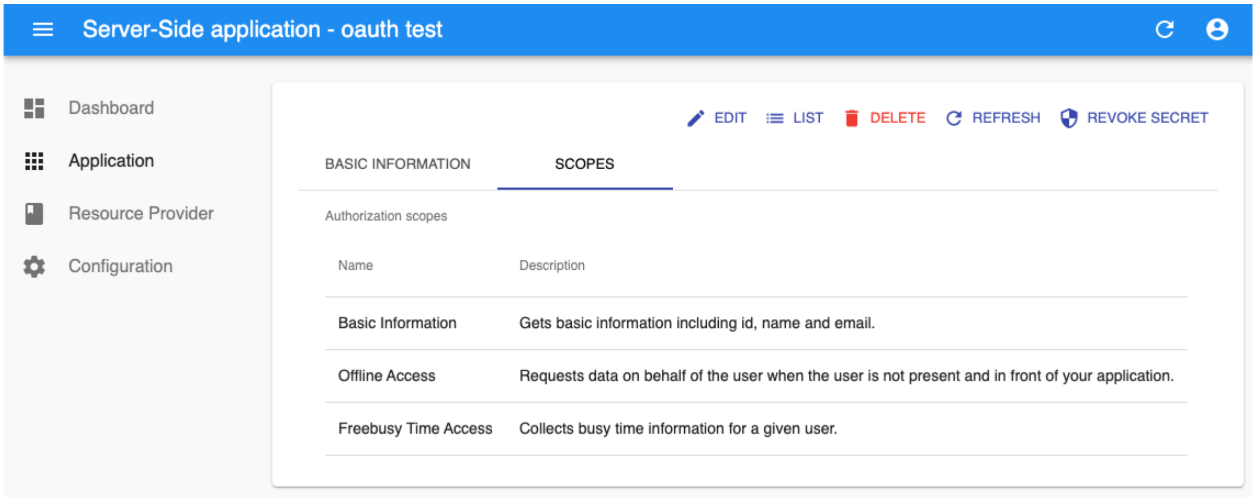
Proton accesses

☑ Domino Database Access

Functional ID (LDAP format)

cn=app1,o=Harbour-Light,c=CA

On the “Scopes” page, select a couple of scopes like so:



Authorization scopes

Name	Description
Basic Information	Gets basic information including id, name and email.
Offline Access	Requests data on behalf of the user when the user is not present and in front of your application.
Freebusy Time Access	Collects busy time information for a given user.

Then, save your application. Return to the Basic Information page and note the Application Id and the Application secret. With this, log out of the Admin interface of the IAM Service.

Next, switch to the sample code in the AppDevPacks IAM Client Package:

```
cd node-iam-client/examples/authorization_code_flow
```

Start Visual Studio code by entering `code .`

First, we have to update our config.js file to reflect the settings for our IAM infrastructure and app to use:

Here’s my sample:

```
/* © Copyright HCL Technologies Ltd. 2018, 2019. All Rights Reserved. */
module.exports = {
// The IAM server.
iam_server: '<your-server>:9443',


// The filepath of IAM server's ca certificate, need to be set when iam cert
is issued by an untrusted CA - copy the technical user of the IAM server !
iam_server_ca_cert: 'iam-service/config/certs/IAMUser.crt',


// The client identifier and client secret issued to the client during the
registration process - Here’s our application id and application secret from
the previous step.
client_id: '182f5724-ee6b-4215-9f88-89102a08c3bd',
client_secret: 'DyIQIh5p0VyMQrw0MUQndd1KkMeMgBq5p5jrCPA/lvzsgo9GMI-
zej9GTmhefckUC',


// The redirection endpoint. After a user successfully authorizes an applica-
tion,
// the authorization server will redirect the user back to the application
with
// either an authorization code or access token in the URL.
redirect_uri: 'https://localhost:3002/cb',

// Scopes the client want to access.
scope: 'openid offline_access domino.proton.db.access das.calendar.read.with.shared das.freebusy',


// The client application port
port: 3002,
};
```

Next, we will define our server configuration for domino-db. Therefore, we create a new file called
server-config.js
Here’ the content:

```
var fs = require('fs');
var path = require('path');
const readFile = fileName => {
try {
return fs.readFileSync(path.resolve(fileName));
} catch (error) {
console.log(error);
return undefined;
}
};
const rootCertificate = readFile('./examples/authorization_code_flow/certificates/ca.crt');
const clientCertificate = readFile('./examples/authorization_code_flow/certificates/app1.crt');
const clientKey = readFile('./examples/authorization_code_flow/certificates/app1.key');
const { useServer } = require('@domino/domino-db');
const serverConfig = {
hostName: '<your-proton-hostname>', // DNS (!) Host name of your server
// See scripts to create kyr-file and ca for adoption !
connection: {
port: '3002', // Proton port on your server
secure: true,
},
credentials: {
rootCertificate,
clientCertificate,
clientKey
}
};
module.exports = serverConfig;
```

This defines the technical user for domino-db as well as the Domino Hostname for the proton task as well as the certificates needed to use ssl when accessing the proton task.

Next, we alter our `index.js` file.

Step 1: Make use of domino-db and the server.config.
copy domino-domino-db-1.3.0.tgz into the same directories as index.js
At the top section add two lines:

```
const {useServer} = require('@domino/domino-db');
const serverConfig = require('./server-config');
```

The first one imports domino-db, the second one imports the domino server config.
Next, we will create a function to handle our Domino call to create new documents in a database:
This is a asynchronous function ad we will pass it a property called access token, that we will get from the authorization process in a minute.
Copy this code as new function into your index.js file:

```
async function createDocumentsinDomino(accesstoken) {
return new Promise(function(resolve,reject){
var back = '';
const databaseConfig = {
filePath: '<your-db>.nsf', // The database file name
```

```
};
// Here, we define the documents we want to create
const createOptions = {
documents: [
{
Form: 'Contact',
FirstName: 'Heiko',
LastName: 'Voigt',
City: 'Shelburne',
State: 'NS',
},
{
Form: 'Contact',
FirstName: 'Graham',
LastName: 'Acres',
City: 'Vancouver',
State: 'BC',
},
],
accessToken: accesstoken,
};


/*
 Now, let's add a document to a Notes Database as a user !
 */
useServer(serverConfig).then(async server => {
const database = await server.useDatabase(databaseConfig);
const response = await database.bulkCreateDocuments(createOptions);


// Display the new document UNIDs
const unids = response.documents.map(doc => doc['@unid']);
console.log(`Documents created: ${unids}`);
back = `Documents created: ${unids}`;
resolve(back);
})
})
}
```

Then, we have to alter our “callback” function to this:

```
const callback = async (ctx) => {
const { secureCtx } = ctx.session;
delete ctx.session.secureCtx;
// get IDToken, refreshToken, and AccessToken
const tokenSet = await iamClient.getToken(ctx.request, secureCtx);
ctx.session.tokenSet = tokenSet;
// decode id token
const idTokenInfo = getIDTokenInfo(tokenSet.id_token);
// add introspection result.
const accessTokenIntrospectionResult = await iamClient.introspectAccessToken(tokenSet.access_token);
const refreshTokenIntrospectionResult = await iamClient.introspectRefreshToken(tokenSet.refresh_token);
// Here’s our new function call
var unidsback = await createDocumentsinDomino(tokenSet.access_token);
await ctx.render('index', {
idTokenInfo,
accessToken: tokenSet.access_token,
refreshToken: tokenSet.refresh_token,
accessTokenIntrospectionResult: JSON.stringify(accessTokenIntrospectionResult),
refreshTokenIntrospectionResult: JSON.stringify(refreshTokenIntrospectionResult),
documentUniqueIds: unidsback,
});
};
```

To finish this, we have to alter the view to display some results:

Open views/index.html. Here, search for `refreshTokenIntrospectionResult`.

Beneath that, you will see an `<hr />` tag. Underneath that, copy the next lines:

```
<pre><b>Domino-DB created Documents</b></pre>
<%= documentUniqueIds %>
<hr />
```

Then, save all open files and start the app from the console in `../iam-client/examples` :

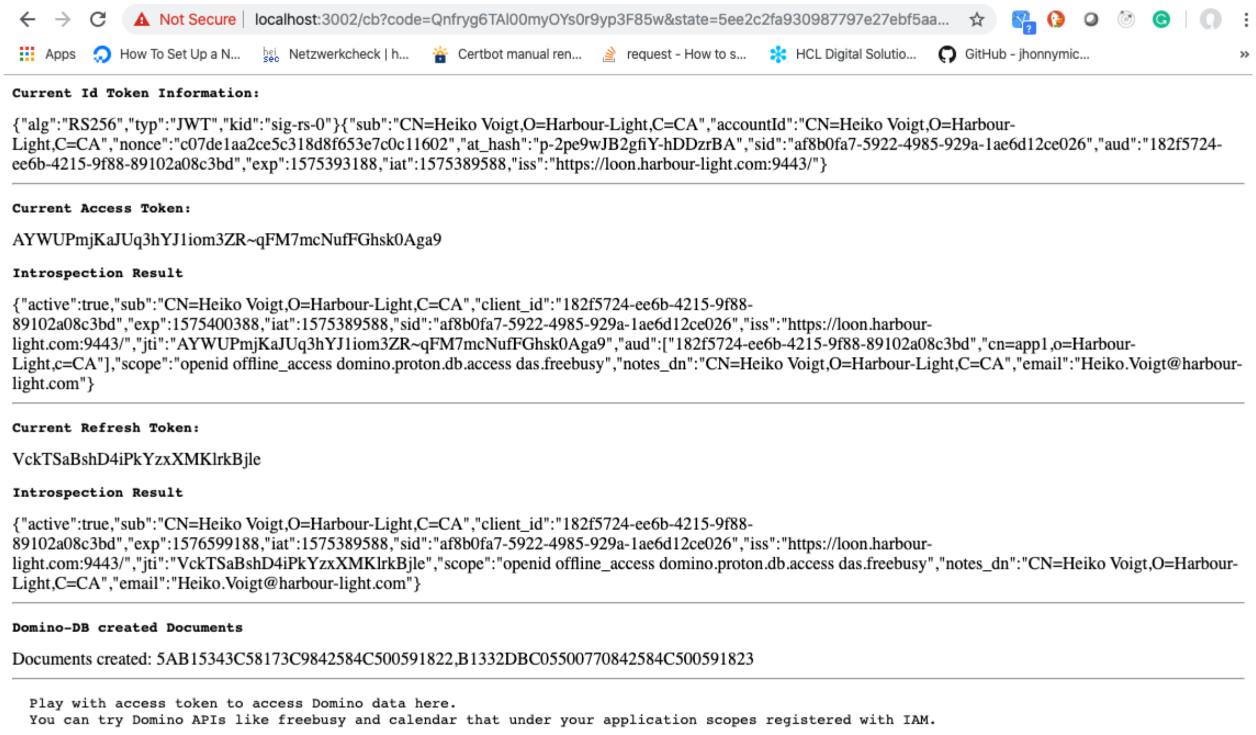
```
npm run auth_code_flow
```

You should see:

```
Application starts, please open https://localhost:3002 with Chrome/Firefox/Safari
```

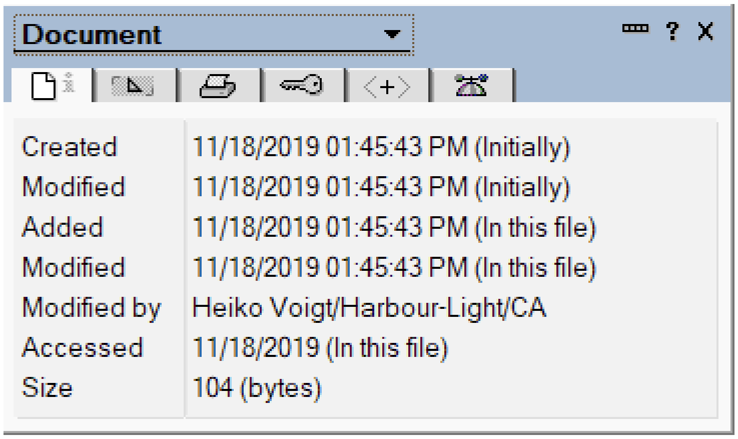
Open your local browser and go to https://localhost:3002

Log in - and after successful log in, you will see a similar output:



```
Current Id Token Information:

{"alg":"RS256","typ":"JWT","kid":"sig-rs-0"}{"sub":"CN=Heiko Voigt,O=Harbour-Light,C=CA","accountId":"CN=Heiko Voigt,O=Harbour-
Light,C=CA","nonce":"c07de1aa2ce5c318d8f653e7c0c11602","at_hash":"p-2pe9wJB2gfiY-hDDzrBA","sid":"af8b0fa7-5922-4985-929a-1ae6d12ce026","aud":"182f5724-
ee6b-4215-9f88-89102a08c3bd","exp":1575393188,"iat":1575389588,"iss":"https://loon.harbour-light.com:9443/"}

Current Access Token:

AYWUPmjKaJUq3hYJ1iom3ZR~qFM7mcNufFGhsk0Aga9

Introspection Result

{"active":true,"sub":"CN=Heiko Voigt,O=Harbour-Light,C=CA","client_id":"182f5724-ee6b-4215-9f88-
89102a08c3bd","exp":1575400388,"iat":1575389588,"sid":"af8b0fa7-5922-4985-929a-1ae6d12ce026","iss":"https://loon.harbour-
light.com:9443/","jti":"AYWUPmjKaJUq3hYJ1iom3ZR~qFM7mcNufFGhsk0Aga9","aud":["182f5724-ee6b-4215-9f88-89102a08c3bd","cn=app1,o=Harbour-
Light,c=CA"],"scope":"openid offline_access domino.proton.db.access das.freebusy","notes_dn":"CN=Heiko Voigt,O=Harbour-Light,C=CA","email":"Heiko.Voigt@harbour-
light.com"}

Current Refresh Token:

VckTSaBshD4iPkYzxXMKlrkBjle

Introspection Result

{"active":true,"sub":"CN=Heiko Voigt,O=Harbour-Light,C=CA","client_id":"182f5724-ee6b-4215-9f88-
89102a08c3bd","exp":1576599188,"iat":1575389588,"sid":"af8b0fa7-5922-4985-929a-1ae6d12ce026","iss":"https://loon.harbour-
light.com:9443/","jti":"VckTSaBshD4iPkYzxXMKlrkBjle","scope":"openid offline_access domino.proton.db.access das.freebusy","notes_dn":"CN=Heiko Voigt,O=Harbour-
Light,C=CA","email":"Heiko.Voigt@harbour-light.com"}

Domino-DB created Documents

Documents created: 5AB15343C58173C9842584C500591822,B1332DBC05500770842584C500591823

  Play with access token to access Domino data here.
  You can try Domino APIs like freebusy and calendar that under your application scopes registered with IAM.
```

If you check the newly created documents in your test database, they should be modified by the user that has logged in using IAM:



Created	11/18/2019 01:45:43 PM (Initially)
Modified	11/18/2019 01:45:43 PM (Initially)
Added	11/18/2019 01:45:43 PM (In this file)
Modified	11/18/2019 01:45:43 PM (In this file)
Modified by	Heiko Voigt/Harbour-Light/CA
Accessed	11/18/2019 (In this file)
Size	104 (bytes)

Well Done, that concludes the basic setup of IAM, OAUTH-DSAPI and the Act-As-User Feature !