pyAS2 Documentation

Release 0.4.2

Abhishek Ram

Jun 09, 2018

Contents

pyAS2 is an AS2 server/client written in python and built on the Django framework. The application supports AS2 version 1.2 as defined in the RFC 4130. Our goal is to provide a native python library for implementing the AS2 protocol. It supports Python 2.6-2.7.

The application includes a server for receiving files from partners, a front-end web interface for configuration and monitoring, a set of django-admin commands that serves as a client for sending messages, asynchronous MDNs and a daemon process that monitors directories and sends files to partners when they are placed in the partner's watched directory.

CHAPTER 1

Features

- Technical
  - Asynchronous and synchronous MDN
  - Partner and Organization management
  - Digital signatures
  - Message encryption
  - Secure transport (SSL)
  - Support for SSL client authentication
  - System task to auto clear old log entries
  - Data compression (AS2 1.1)
  - Multinational support: Uses Django's internationalization feature
- Integration
  - Easy integration to existing systems, using a partner based file system interface
  - Daemon Process picks up data from directories when it becomes available
  - Message post processing (scripting on receipt)
- Monitoring
  - Web interface for transaction monitoring
  - Email event notification
- The following encryption algorithms are supported:
  - Triple DES
  - DES
  - RC2-40
  - AES-128

  - AES-192
  - AES-256
- The following hash algorithms are supported:
  - SHA-1

CHAPTER 2

Dependencies

- Python (2.6.5+, 2.7+)
- Django (1.7+)
- M2Crypto (This is dependent on openssl.)
- requests
- pyasn1
- cherrypy
- pyinotify on *nix (Optional for using the `send daemon`)
- Python for Windows extensions (pywin) for windows (Optional for using the `send daemon`)

CHAPTER 3

Installation

You can install `pyAS2` with `pip`:

```
$ pip install pyas2
```

See *Installation* for more information.

CHAPTER 4

Table of Contents:

4.1 Installation

Fisrt Install M2Crypto separately following these instructions

Install using `pip`...

```
$ pip install pyas2
```

Create a new `django` project

```
$ django-admin.py startproject django_pyas2
```

Add `pyas2` to your `INSTALLED_APPS` setting, ensure that `pyas2` is placed at the top of this list.

```
INSTALLED_APPS = (
    'pyas2',
    ...
)
```

Include the pyAS2 URL configuration in your project's `urls.py`.

```
from django.conf.urls import include #add only if django version >= 1.9
url(r'^pyas2/', include('pyas2.urls')),
```

Run the following commands to complete the installation and start the server.

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: pyas2, admin, contenttypes, auth, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
```

(continues on next page)

(continued from previous page)

```
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying pyas2.0001_initial... OK
  Applying pyas2.0002_partner_compress... OK
  Applying pyas2.0003_auto_20150311_1141... OK
  Applying pyas2.0004_auto_20150311_1258... OK
  Applying pyas2.0005_message_compressed... OK
  Applying pyas2.0006_auto_20150313_0548... OK
  Applying pyas2.0007_auto_20150313_0707... OK
  Applying pyas2.0008_auto_20150317_0450... OK
  Applying pyas2.0009_auto_20150317_1324... OK
  Applying pyas2.0010_auto_20150416_0745... OK
  Applying pyas2.0011_auto_20150427_1029... OK
  Applying pyas2.0012_auto_20151006_0526... OK
  Applying pyas2.0013_auto_20160307_0233... OK
  Applying pyas2.0014_auto_20160420_0515... OK
  Applying pyas2.0015_auto_20160615_0409... OK
  Applying pyas2.0016_auto_20161004_0543... OK
  Applying sessions.0001_initial... OK
$ python manage.py createsuperuser
Username (leave blank to use 'abhishekram'): admin
Email address: admin@domain.com
Password:
Password (again):
Superuser created successfully.
$ python manage.py runas2server
20150908 07:14:32 Level 25 : PyAS2 server running at port: "8080".
20150908 07:14:32 Level 25 : PyAS2 server uses plain http (no ssl).
```

The pyAS2 server is now up and running, the web UI for configuration and monitoring can be accessed at `http://{hostname}:8080/pyas2/` and the endpoint for receiving AS2 messages from your partners will be at `http://{hostname}:8080/pyas2/as2receive`

4.1.1 Upgrading pyAS2

Upgrading to the latest version of pyAS2 is a straight forward procedure. We will use `pip` to update the package to the latest version and django's migrations framework to migrate the database to reflect any changes made to the models.

Run the following commands to upgrade to the latest version:

```
$ pip install -U pyas2
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, pyas2, contenttypes, auth, sessions
Running migrations:
  Applying pyas2.0017_auto_20170404_0730... OK
```

Warning: A major change has been made to pyAS2, starting version 0.3.4 the migrations are included in the repo so if you are upgrading from an older version you need to fake till the last migration done and then finally do migrations. So suppose you were at 0.3.2 you would follow these steps:

```
$ python manage.py migrate --fake pyas2 0016
Operations to perform:
  Target specific migration: 0016_auto_20161004_0543, from pyas2
Running migrations:
  Applying pyas2.0002_partner_compress... FAKED
  Applying pyas2.0003_auto_20150311_1141... FAKED
  Applying pyas2.0004_auto_20150311_1258... FAKED
  Applying pyas2.0005_message_compressed... FAKED
  Applying pyas2.0006_auto_20150313_0548... FAKED
  Applying pyas2.0007_auto_20150313_0707... FAKED
  Applying pyas2.0008_auto_20150317_0450... FAKED
  Applying pyas2.0009_auto_20150317_1324... FAKED
  Applying pyas2.0010_auto_20150416_0745... FAKED
  Applying pyas2.0011_auto_20150427_1029... FAKED
  Applying pyas2.0012_auto_20151006_0526... FAKED
  Applying pyas2.0013_auto_20160307_0233... FAKED
  Applying pyas2.0014_auto_20160420_0515... FAKED
  Applying pyas2.0015_auto_20160615_0409... FAKED
  Applying pyas2.0016_auto_20161004_0543... FAKED
$ python manage.py migrate pyas2
Operations to perform:
  Apply all migrations: pyas2
Running migrations:
  Applying pyas2.0017_auto_20170404_0730... OK
```

4.2 Configuration

The global settings for pyAS2 are kept in a single configuration dictionary named PYAS2 in your project's settings.py module. Below is a sample configuration:

```
PYAS2 = {
    'ENVIRONMENT' : 'production',
    'PORT' : 8888,
    'SSLCERTIFICATE' : '/path_to_cert/server_cert.pem',
    'SSLPRIVATEKEY' : '/path_to_cert/server_privkey.pem',
    'DATADIR' : '/path_to_datadir/data',
    'PYTHONPATH' : '/path_to_python/python',
    'ENVIRONMENTTEXT' : 'BETA',
    'ENVIRONMENTTEXTCOLOR' : 'Yellow',
    'LOGLEVEL' : 'DEBUG',
    'LOGCONSOLE' : True,
    'LOGCONSOLELEVEL' : 'DEBUG',
    'MAXRETRIES': 5,
    'MDNURL' : 'https://192.168.1.115:8888/pyas2/as2receive',
    'ASYNCMDNWAIT' : 30,
    'MAXARCHDAYS' : 30,
}
```

The available settings along with their usage is described below:

Settings Name	Default Value	Usage
ENVIRONMENT	production	The as2 server in development or production
PORT	8080	HTTP Port as2 server listens on
SSLCERTIFICATE	`None`	Path to the SSL Public Key
SSLPRIVATEKEY	`None`	Path to the SSL Private Key
DATADIR	`Django Project Path`	Full path to the base directory for storing messages, MDNs, certificates and logs
PYTHONPATH	`System Python Path`	Path to the python executable, required with virtual environments
ENVIRONMENT-TEXT	`None`	Text displayed on right of the logo. Useful to indicate different environments.
ENVIRONMENT-TEXTCOLOR	Black	Color of the displayed PYTHONPATH. Use HTML valid "color name" or #RGB values.
LOGLEVEL	INFO	Level for logging to log file. Values: DEBUG,INFO,STARTINFO,WARNING,ERROR or CRITICAL.
LOGCONSOLE	True	Console logging on (True) or off (False).
LOGCONSOLELEVEL	STARTINFO	level for logging to console/screen. Values: DEBUG,INFO,STARTINFO,WARNING,ERROR or CRITICAL.
MAXRETRIES	10	Maximum number of retries for failed outgoing messages
MDNURL	`None`	Return URL for receiving asynchronous MDNs from partners.
ASYNCMDNWAIT	30	Number of minutes to wait for asynchronous MDNs after which message will be marked as failed.
MAXARCHDAYS	30	Number of days files and messages are kept in storage.

4.3 Quick-start Guide

Now that we have completed installation and configuration of pyAS2, we are ready to start transferring files.

Let's get started by sending a signed and encrypted file from one pyAS2 server `P1` to another pyAS2 server `P2`. Do note that these two are separate installations of pyAS2.

4.3.1 Installing the Servers

Create a Django project called `P1` and follow the *installation guide* and run `python manage.py runas2server` to start `P1` at http://localhost:8080/pyas2/

pyAS2 Server Configurations

Current user	admin
Last login	2015-09-08 7:19
Environment	P1
Python Path	/Users/abhishekram/Documents/work/Research/pythondev/bin/python
Manage.py Path	/Users/abhishekram/Documents/work/Research/pyas2_test/P1/manage.py
Base Data Dir	/Users/abhishekram/Documents/work/Research/pyas2_test/P1
ASYNC MDN Receive URL	http://localhost:8080/pyas2/as2receive
ASYNC MDN Max Wait Mins	30
Max Retries	30
Log Folder	/Users/abhishekram/Documents/work/Research/pyas2_test/P1/logging
Log Level	INFO
Log Console	True
Log Console Level	STARTINFO
Max Archive Days	30

Create one more Django project called `P2` and follow the same installations instructions, however now we will need to change the pyAS2 port as `P1` is using the default port. To do this update the port to 8081 in the *global settings* and run `python manage.py runas2server` to start `P2` at http://localhost:8081/pyas2/

pyAS2 Server Configurations

Current user	admin
Last login	2015-09-08 7:32
Environment	P2
Python Path	/Users/abhishekram/Documents/work/Research/pythondev/bin/python
Manage.py Path	/Users/abhishekram/Documents/work/Research/pyas2_test/P2/manage.py
Base Data Dir	/Users/abhishekram/Documents/work/Research/pyas2_test/P2
ASYNC MDN Receive URL	http://localhost:8080/pyas2/as2receive
ASYNC MDN Max Wait Mins	30
Max Retries	30
Log Folder	/Users/abhishekram/Documents/work/Research/pyas2_test/P2/logging
Log Level	INFO
Log Console	True
Log Console Level	STARTINFO
Max Archive Days	30

4.3.2 Creating the certificates

We need to generate a Public and Private key pair each for the two servers. `P1` uses its private key to sign the message which is verified by `P2` using `P1's` public key. `P1` uses the `P2's` public key to encrypt the message which is decrypted by `P2` using its private key.

To generate the public and private key pair use the below commands

```
$ openssl req -x509 -newkey rsa:2048 -keyout P1_private.pem -out P1_public.pem -days␣
↪365
Generating a 2048 bit RSA private key
.....+++
.....
↪..........+++
writing new private key to 'P1_private.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
```

(continues on next page)

(continued from previous page)

```
State or Province Name (full name) [Some-State]:Karnataka
Locality Name (eg, city) []:Bangalore
Organization Name (eg, company) [Internet Widgits Pty Ltd]:P1
Organizational Unit Name (eg, section) []:AS2
Common Name (e.g. server FQDN or YOUR name) []:p1as2
Email Address []:
$ cat P1_public.pem >> P1_private.pem

$ openssl req -x509 -newkey rsa:2048 -keyout P2_private.pem -out P2_public.pem -days␣
↪365
Generating a 2048 bit RSA private key
..............................+++
.....................................................................................
↪.....................+++
writing new private key to 'P2_private.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Karnataka
Locality Name (eg, city) []:Bangalore
Organization Name (eg, company) [Internet Widgits Pty Ltd]:P2
Organizational Unit Name (eg, section) []:AS2
Common Name (e.g. server FQDN or YOUR name) []:p2as2
Email Address []:
$ cat P2_public.pem >> P2_private.pem
```

4.3.3 Configure P1

`P1` needs to be configured before it can start sending files, open the web UI and follow these instructions:

- Navigate to `Configuration->Private Certificates->Add private certifcate`.
- Choose the file `P1_private.pem` in the certificate field, enter the passphrase and save the Private Certificate.
- Next navigate to `Configuration->Public Certificates->Add public certifcate`.
- Choose the file `P2_public.pem` in the certificate field and save the Public Certificate.
- Now navigate to `Configuration->Organization->Add organization`.
- Set Name to `P1`, As2 Name to `p1as2` and set the Signature and Encryption keys to `P1_private.pem` and save the Organization.
- Next navigate to `Configuration->Partner->Add partner`.
- Set Name to `P2`, As2 Name to `p2as2` and Target url to `http://localhost:8081/pyas2/as2receive`
- Under security settings set Encrypt Message to `3des`, Sign Message to `SHA-1`, Signature and Encryption keys to `P2_public.pem`.

- Under MDN settings set MDN mode to `Synchronous` and Request Signed MDN to `SHA-1`.
- Save the partner to complete the configuration.

4.3.4 Configure P2

`P2` needs to be configured before it can start receiving files, open the web UI and follow these instructions:

- Navigate to `Configuration->Private Certificates->Add private certifcate`.
- Choose the file `P2_private.pem` in the certificate field, enter the passphrase and save the Private Certificate.
- Next navigate to `Configuration->Public Certificates->Add public certifcate`.
- Choose the file `P1_public.pem` in the certificate field and save the Public Certificate.
- Now navigate to `Configuration->Organization->Add organization`.
- Set Name to `P2`, As2 Name to `p2as2` and set the Signature and Encryption keys to `P2_private.pem` and save the Organization.
- Next navigate to `Configuration->Partner->Add partner`.
- Set Name to `P1`, As2 Name to `p1as2` and Target url to `http://localhost:8080/pyas2/as2receive`
- Under security settings set Encrypt Message to `3des`, Sign Message to `SHA-1`, Signature and Encryption keys to `P1_public.pem`.
- Under MDN settings set MDN mode to `Synchronous` and Request Signed MDN to `SHA-1`.
- Save the partner to complete the configuration.

4.3.5 Send a File

We are now read to send a file from `P1` to `P2`, to do so follow these steps:

- Open the `P1` web UI and navigate to `Run->Send Message`.
- Select the Organization as `p1as2(P1)` and Partner as `p2as2(P2)`.
- Now select the file to send and click `Send File`.
- The status of the file transfer can be viewed at `Messages->All Messages`.
- Once file transfer is completed you will a green tick in the status column.

- We will also see a similar entry in the web UI of `P2`.

- We can see basic information on this screen such as Partner, Organization, Message ID and MDN.
- We can also view the MDN and Payload by clicking on the respective links.

4.3.6 Conclusion

We have successfully demonstrated the core functionality of pyAS2 i.e. sending files from one system to another using the AS2 protocol. For a more detailed overview of all its functionality do go through the *detailed docs*.

4.4 Detailed Guide

We have seen how to send a file to the partner with the basic settings. Now lets go through each of the components of pyAS2 in greater detail. In this section we will cover topics related to configuration of partners, organizations and certificates; sending messages and MDNs; monitoring messages and MDNs; and usage of the admin commands.

4.4.1 Organizations

Organizations in pyAS2 mean the host of the AS2 server, i.e. it is the sender when sending messages and the receiver when receiving the messages. Organizations can be managed from the web UI at Configuration->Organizations. The following screen lists the existing organizations and also you gives the option to create new ones. Each organization is characterized by the following fields:

Field Name	Description	Mandatory
Name	The descriptive name of the organization.	Yes
As2 Name	The as2 identifies for this organization, must be a unique value as it identifies the as2 host.	Yes
Email Address	The email address for the organization.	No
Encryption Key	The Private Key used for decrypting incoming messages from trading partners.	Yes
Signature Key	The Private Key used to sign outgoing messages to trading partners	Yes

4.4.2 Partners

Partners in pyAS2 mean all your trading partners with whom you will exchanges messages, i.e. they are the receivers when you send messages and the senders when you receive messages. Partners can be managed from the web UI at Configuration->Partners. The following screen lists the existing partners and also you gives the option to search them and create new ones. Each partner is characterized by the following fields:

General Settings

Field Name	Description	Mandatory
`Name`	The descriptive name of the partner.	Yes
`As2 Name`	The as2 identifies for this partner as communicated by the partner.	Yes
`Email Address`	The email address for the partner.	No
`Target Url`	The HTTP/S endpoint of the partner to which files need to be posted.	Yes
`Subject`	The MIME subject header to be sent along with the file.	Yes
`Content Type`	The content type of the message being transmitted, can be XML, X12 or EDIFACT.	Yes

Authentication Settings

Use these settings if basic authentication has been enabled for the partners AS2 server.

Field Name	Description	Mandatory
`Enable Authentication`	Check this option to enable basic AUTH.	No
`Http auth user`	User name to access the partners server.	No
`Http auth pass`	Password to access the partners server.	No
`HTTPS Local CA Store`	Use this for HTTPS endpoints where the partners SSL certificate has been signed by an unknown CA. Select the CA certificate here	No

Security Settings

Field Name	Description	Mandatory
`Compress Message`	Check this option to enable AS2 message compression.	Yes
`Encrypt Message`	Select the algorithm to be used for encrypting messages, defaults to None.	No
`Encryption Key`	Select the `Public Key` used for encrypting the outbound messages to this partner.	No
`Sign Message`	Select the hash algorithm to be used for signing messages, defaults to None. incoming messages from trading partners.	No
`Signature key`	The `Public Key` used to verify inbound signed messages and MDNs from this partner	No

MDN Settings

Field Name	Description	Mandatory
`Request MDN`	Check this option to request MDN for outbound messages to this partner.	Yes
`Mdn mode`	Select the MDN mode, defaults to Synchronous	No
`Request Signed MDN`	Select the algorithm to be used in case signed MDN is to be returned.	No

Advanced Settings

Field Name	Description	Mandatory
`Keep Original Filename`	Use Original File name to to store file on receipt, use this option only if you are sure partner sends unique names.	No
`Command on Message Send`	OS Command executed after successful message send, replacements are `$filename`, `$sender`, `$recevier`, `$messageid` and any message header such as `$Subject`	No
`Command on Message Receipt`	OS Command executed after successful message receipt, replacements are `$filename`, `$fullfilename`, `$sender`, `$recevier`, `$messageid` and any message header such as `$Subject`.	No

4.4.3 Certificates

The AS2 protocol strongly encourages the use of RSA certificates to sign and encrypt messages for enhanced security. A signed and encrypted message received from your partner ensures message repudiation and integrity. The RSA certificate consists of a public key and a private key which are together used for encrypting, decrypting, signing and verifying messages.

Generating Certificates

When you set up a new AS2 server you will need to generate a Public/Private key pair. The private key will be added to your server and the public key needs to be shared with your trading partners.

One of the ways of generating a certificate is by using the `openssl` command line utility, the following command needs to be used:

```
$ openssl req -x509 -newkey rsa:2048 -keyout private.pem -out public.pem -days 365
Generating a 2048 bit RSA private key
.....+++
.....
↪.....+++
writing new private key to 'private.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

(continues on next page)

(continued from previous page)

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Karnataka
Locality Name (eg, city) []:Bangalore
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Name
Organizational Unit Name (eg, section) []:AS2
Common Name (e.g. server FQDN or YOUR name) []:as2id
Email Address []:
$ cat public.pem >> private.pem
```

The above commands will generate a PEM encoded private key called `private.pem` and a PEM encoded public key called `public.pem`.

Private Certificates

`Private Certificates` are used for signing outbound messages to your partners and decrypting incoming messages from your partners. We can manage them in `pyAS2` from the web UI at `Configuration->Private Certificates`. The following screen lists all your private certificates and lets you add new ones. Each `Private Certificates` is characterized by the following fields:

Field Name	Description	Mandatory
`Certificate`	Select the **PEM** encoded[1] private key file[2].	Yes
`Local CA Store`	In case the certificate has been signed by an unknown CA then select the CA certificate here.	No
`Certificate passphrase`	The pass phrase entered at the time of the certificate generation.	Yes

Public Certificates

`Public Certificates` are used for verifying signatures of inbound messages and encrypting outbound messages to your partners. The public key file will be shared by your partner. We can manage them in `pyAS2` from the web UI at `Configuration->Public Certificates`. The following screen lists all your public certificates and lets you add new ones. Each `Public Certificates` is characterized by the following fields:

Field Name	Description	Mandatory
`Certificate`	Select the **PEM** encoded[1] public key file.	Yes
`Local CA Store`	In case the certificate has been signed by an unknown CA then select the CA certificate here.	No
`Verify Certificate` New in version 0.2	Uncheck this option to disable certificate verification at the time of signature verification.	No

[1] `pyAS2` supports only PEM encoded certificates.
[2] The private key file must contain **both the private and public** parts of the RSA certificate.

4.4.4 The Data Directory

The `Data Directory` is a file system directory that stores certificates, logs, archives, sent and received files. The location of this directory is set to the `Django` project folder by default. We can also change this directory by updating the `DATADIR` setting in the `pyAS2` *global settings*. The structure of the directory is below:

```
{DATA DIRECTORY}
├── certificates
│   ├── P1_public.pem
│   └── P2_private.pem
├── logging
│   ├── cherrypy_error.log
│   ├── pyas2.log
│   └── pyas2.log.2015-09-08
└── messages
    ├── __store
    │   ├── mdn
    │   │   ├── received
    │   │   └── sent
    │   │       ├── 20150908
    │   │       │   ├── 20150908115337.7244.44635@Abhisheks-MacBook-Air.local.mdn
    │   │       │   └── 20150908121942.7244.71894@Abhisheks-MacBook-Air.local.mdn
    │   │       └── 20150913
    │   │           ├── 20150913071324.20065.47671@Abhisheks-MacBook-Air.local.mdn
    │   │           └── 20150913083125.20403.32480@Abhisheks-MacBook-Air.local.mdn
    │   └── payload
    │       ├── received
    │       │   ├── 20150908
    │       │   │   ├── 20150908115458.7255.98107@Abhisheks-MacBook-Air.local
    │       │   │   └── 20150908121933.7343.83150@Abhisheks-MacBook-Air.local
    │       │   └── 20150913
    │       │       ├── 20150913071323.20074.48016@Abhisheks-MacBook-Air.local
    │       │       └── 20150913083125.20475.14667@Abhisheks-MacBook-Air.local
    │       └── sent
    ├── p1as2
    │   └── outbox
    │       └── p2as2
    └── p2as2
        └── inbox
            └── p1as2
                ├── 20150908115458.7255.98107@Abhisheks-MacBook-Air.local.msg
                └── 20150913083125.20475.14667@Abhisheks-MacBook-Air.local.msg
```

certificates

The `certificates` directory stores all the `PEM` encoded public and private key files.

logging

The `logging` directory stores the server error logs and application logs. The server error logs are saved as `cherrypy_error.log` and the application logs are saved as `pyas2.log`.

__store

The `__store` directory under the `messages` directory archives the payloads and MDNs. The `payloads` directory saves the sent and received files in the corresponding sub-folders and the `mdn` directory also does the same for sent and received MDNs. The payloads and MDNs in the sent or received folders are further saved under sub-folders for each day named as `YYYYMMDD`.

inbox

The inbox directory stores files received from your partners. The path of this directory is `{DATA DIRECTORY}/{ORG AS2 ID}/inbox/{PARTNER AS2 ID}`. We need to take this location into account when integrating `pyAS2` with other applications.

outbox

The outbox folder works in conjecture with the `send-daemon` process. The daemon process monitors all the outbox folder and will trigger a transfer when a file becomes available. The path of this directory is `{DATA DIRECTORY}/{PARTNER AS2 ID}/outbox/{ORG AS2 ID}`.

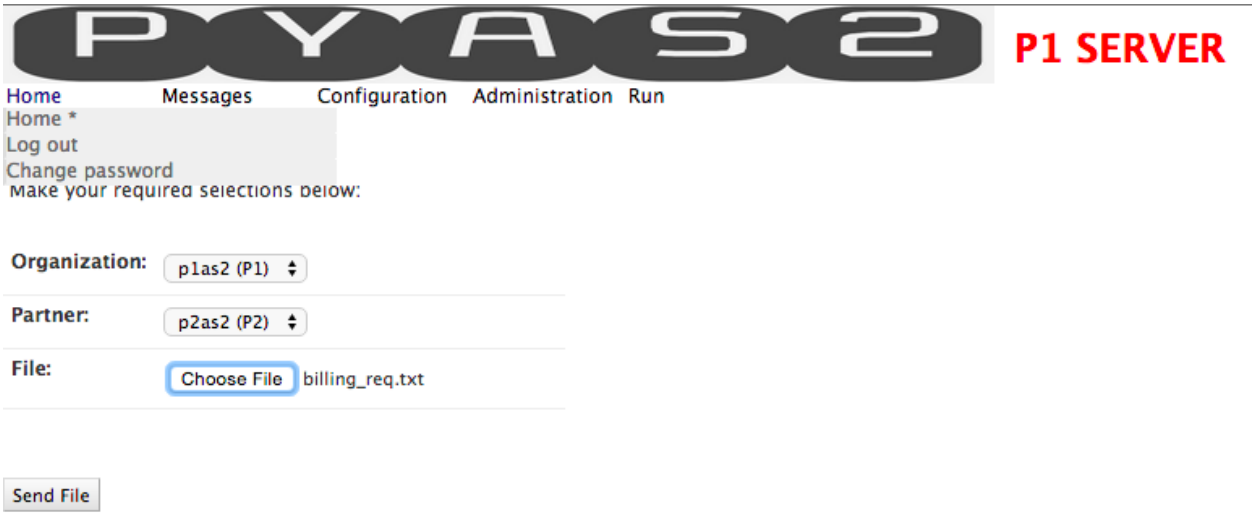
4.4.5 Send & Receive Messages

We have so far covered all the topics related to configuration of the `pyAS2` server. Now we will see how to use these configurations to send messages to your trading partners using the AS2 protocol. We can send files using any of the following techniques:

Send Messages From the Web UI

The simplest method for sending messages to your trading partner is by using the Web UI. This method is generally used for testing the AS2 connection with your trading partner. The steps are as follows:

- Navigate to `Run->Send Message`.
- Select the sender(Organization), the receiver(Partner) and choose the file to be transmitted.
- Click on `Send File` to initiate the file transfer and monitor the transfers at `Messages->Outbound Messages`.



Send Messages From the Command-Line

The next method for sending messages involves the `pyAS2` admin command `sendas2message`. The command is invoked from the shell prompt and can be used by other applications to invoke an AS2 file transfer. The command usage is as follows:

```
$ python manage.py sendas2message --help
Usage: python manage.py sendas2message [options] <organization_as2name partner_
↪as2name path_to_payload>

Send an as2 message to your trading partner

Options:
  --delete              Delete source file after processing
  -h, --help            show this help message and exit
```

The mandatory arguments to be passed to the command include `organization_as2name` i.e. the AS2 Identifier of this organization, `partner_as2name` i.e. the AS2 Identifier of your trading partner and `path_to_payload` the full path to the file to be transmitted. The command also lets you set the `--delete` option to delete the file once it is begins the transfer. A sample usage of the command:

```
$ python manage.py sendas2message p1as2 p2as2 /Users/abhishekram/Downloads/
↪updateInvoice.txt
```

Send Messages Using the Send-Daemon

The last method for sending messages involves the `pyAS2` admin command `runas2daemon`. The command once started in the background monitors the data directory and when a file is available in a partner's outbox folder then the transfer is initiated for that file.

```
$ python manage.py runas2daemon
20150915 04:23:37 INFO     : Starting PYAS2 send daemon.
20150915 04:23:37 INFO     : Process exisitng files in the directory.
20150915 04:23:37 INFO     : PYAS2 send daemon started started.
20150915 04:23:37 INFO     : Watching directory /opt/pyapp/djproject/messages/
↪MTSAS2Tst/outbox/pyas2test
20150915 04:23:37 INFO     : Watching directory /opt/pyapp/djproject/messages/
↪likemindsas2/outbox/pyas2test
```

The above example runs the admin command in the foreground, however in a production environment it should be started in the background and also OS specific configuration should be added to start this process on system startup.

Receive Messages

In order to receive files from your trading partners they need to post the AS2 message to the URL `http://{hostname}:{port}/pyas2/as2receive`. The configuration of the *Organization*, *Partner* and *Certificates* need to be completed for successfully receiving messages from your trading partner. Once the message has been received it will be placed in the organizations inbox folder.

4.4.6 Send & Receive MDNs

Message Disposition Notifications or MDNs are return receipts used to notify the sender of a message of any of the several conditions that may occur after successful delivery. In the context of the AS2 protocol, the MDN is used to notify if the message was successfully processed by the receiver's system or not and in case of failures the reason for the failure is sent with the MDN.

MDNs can be transmitted either in a synchronous manner or in an asynchronous manner. The synchronous transmission uses the same HTTP session as that of the AS2 message and the MDN is returned as an HTTP response message. The asynchronous transmission uses a new HTTP session to send the MDN to the original AS2 message sender.

Send MDNs

The choice of whether to send an MDN and its transfer mode is with the sender of the AS2 message. The sender lets us know what to do through an AS2 header field. In case the partner requests a synchronous MDN no action is needed as `pyAS2` takes care of this internally, however in the case of an asynchronous MDN the admin command `sendasyncmdn` needs to be run to send the MDN to the trading partner.

The command `{PYTHONPATH}/python {DJANGOPROJECTPATH}/manage.py sendasyncmdn` should be scheduled every 10 minutes so that `pyAS2` sends any pending asynchronous MDN requests received from your trading partners.

Receive MDNs

The choice of whether or not to receive MDN and its transfer mode is with us. The MDN Settings for the partner should be used to specify your preference. In case of synchronous mode `pyAS2` processes the received MDN without any action from you.

In the case of asynchronous mode we do need to take care of a couple of details to enable the receipt of the MDNs. The *global setting* `MDNURL` should be set to the URL `http://{hostname}:{port}/pyas2/as2receive` so that the trading partner knows where to send the MDN. The other setting of note here is the `ASYNCMDNWAIT` that decides how long `pyAS2` waits for an MDN before setting the message as failed so that it can be retried. The admin command `sendasyncmdn` makes this check for all pending messages so it must be scheduled to run regularly.

4.4.7 Monitoring

pyAS2 maintains a log of all inbound and outbound messages exchanged with your trading partners. The logs can be accessed from the web UI Messages menu. The menu has options to list messages, search messages and MDNs. pyAS2 saves message details such as status, message ID, sender, receiver, payload; and MDN details such as message ID, original message ID and mode.

List Messages

The list of all sent and received messages can be viewed from the web UI at Messages->All Messages. The screen lists all messages ordered by timestamp so that the latest message is first on the list. We can further list only inbound messages at Messages->Inbound Messages and outbound messages at Messages->Outbound Messages.

Search Messages

pyAS2 lets you search for messages based on a number of criteria. The search screen can be accessed at Messages->Search Messages. The following filter criteria are available:

Field Name	Description
Datefrom	Messages processed after this date will be included in the search results.
Dateuntil	Messages processed before this date will be included in the search results.
Organization	Messages that belong to this organization will be included.
Partner	Messages that belong to this partner will be included.
Direction	Filter by the direction of the AS2 message, can be inbound or outbound.
Status	Filter by the status of the AS2 message.
Message ID	Filter by the AS2 message ID of the message.
Payload Name	Filter by the file name of the sent/received message.

Search MDNs

pyAS2 also lets you search for MDNs for messages based on a number of criteria. The search screen can be accessed at Messages->Search MDNs. The following filter criteria are available:

Field Name	Description
Datefrom	MDNs processed after this date will be included in the search results.
Dateuntil	MDNs processed before this date will be included in the search results.
Organization	MDNs that belong to this organization will be included.
Partner	MDNs that belong to this partner will be included.
MDN mode	Filter by the MDN mode, can be synchronous or asynchronous.
Status	Filter by the status of the MDN.
MDN Message ID	Filter by the message ID of the MDN.
Original Message ID	Filter by the message ID of the original message for which it is an MDN.

4.4.8 Admin Commands

pyAS2 provides a set of Django manage.py admin commands that perform various functions. We have already seen the usage of some of these commands in the previous sections. Let us now go through the list of available commands:

runas2server

The `runas2server` command starts the AS2 server which includes both the web UI and the AS2 receiver. The command does not take any arguments. The command should be started in the background and also a schedule should be added to run the command on system startup.

runas2daemon

The `runas2daemon` command starts the directory monitoring process. The process monitors all the partner inbox folders and triggers a file transfer when file becomes available. The command should be started in the background and also a schedule should be added to run the command on system startup. The process needs to be restarted when a new partner is created so that its inbox can be added to the monitored directory list.

sendas2message

The `sendas2message` command triggers a file transfer, it takes the mandatory arguments organization id, partner id and the full path to the file to be transferred. The command can be used by other applications to integrate with `pyAS2`.

sendasyncmdn

The `sendasyncmdn` command performs two functions; it sends asynchronous MDNs for messages received from your partners and also checks if we have received asynchronous MDNs for sent messages so that the message status can be updated appropriately. The command does not take any arguments and should be run on a repeating schedule.

retryfailedas2comms

The `retryfailedas2comms` command checks for any messages that have been set for retries and then retriggers the transfer for these messages. The command does not take any arguments and should be run on a repeating schedule.

cleanas2server

The `cleanas2server` command is a maintenance command and it deletes all DB objects, logs and files older that the `MAXARCHDAYS` setting. It is recommended to run this command once a day using cron or windows scheduler.

4.4.9 Email Notifications

We can configure `pyAS2` to send email reports in case of errors encountered while sending or receiving AS2 messages with your trading partner. To use this feature just set the relevent information in your project's settings.py module:

```
MANAGERS = (    #bots will send error reports to the MANAGERS
    ('name_manager', 'myemailaddress@gmail'),
    )
EMAIL_HOST = 'smtp.gmail.com'             #Default: 'localhost'
EMAIL_PORT = '587'             #Default: 25
EMAIL_USE_TLS = True       #Default: False
EMAIL_HOST_USER = 'username'        #Default: ''. Username to use for the SMTP server␣
↪defined in EMAIL_HOST. If empty, Django won't attempt authentication.
EMAIL_HOST_PASSWORD = '*******'    #Default: ''. PASSWORD to use for the SMTP server␣
↪defined in EMAIL_HOST. If empty, Django won't attempt authentication.
```

(continues on next page)

(continued from previous page)

```
SERVER_EMAIL = 'botserrors@gmail.com'           #Sender of bots error reports.␣
↪Default: 'root@localhost'
EMAIL_SUBJECT_PREFIX = ''    #This is prepended on email subject.
```

4.5 Release History

4.5.1 0.4.3 - 2018-06-09

- Send encrypted content as binary data instead of base64

4.5.2 0.4.2 - 2018-06-07

- Also look for *application/x-pkcs7-signature* when verifying MDN signatures
- Limit size of exception logged to database
- Handle case where Sync MDN does not have a Message ID

4.5.3 0.4.1 - 2018-05-07

- Also look for *application/x-pkcs7-signature* when verifying signatures

4.5.4 0.4.0 - 2018-01-27

- Cleaner handling of signature verifications
- Added test cases for sterling b2b integrator message and mdn
- Set *max_length* for file fields to manage long folder names.

4.5.5 0.3.8 - 2018-01-09

- Give option to download certs from the admin.

4.5.6 0.3.7 - 2018-01-09

- Use a function to get the certificate upload_to.

4.5.7 0.3.6 - 2018-01-05

- Added view for downloading certificates from the admin.

4.5.8 0.3.5 - 2017-12-20

- Renewed the certificates used in the django tests.

4.5.9 0.3.4 - 2017-08-17

- Add migration to the distribution.

4.5.10 0.3.3 - 2017-04-04

- Use pagination when listing messages in the GUI, also do not use Datatables.
- Set the request MDN field default value to False.

4.5.11 0.3.2 - 2017-03-07

- Freeze versions of django and CherryPy in setup.py.

4.5.12 0.3.1 - 2016-10-03

- Fixed pagination issue where it was showing only 25 messages and mdns.
- Added the admin command cleanas2server for deleting old data and logs.

4.5.13 0.3.0 - 2016-06-28

- Added django test cases for testing each of the permutations as defined in RFC 4130 Section 2.4.2
- Code now follows the pep-8 standard
- Django admin commands now use argparse instead or optparse

4.5.14 0.2.3 - 2016-04-20

- Added functionality to customize MDN messages at organization and partner levels.

4.5.15 0.2.2 - 2015-10-12

- Fixes to take care of changes in Django 1.9.x

4.5.16 0.2.1 - 2015-10-12

- Updated installation and upgrade documentation.

4.5.17 0.2 - 2015-10-11

- Added option to disable verification of public certificates at the time of signature verification.
- Fixed bug in the send daemon.
- Added debug log statements.
- Added some internationlization to model fields.

4.5.18 0.1.2 - 2015-09-07

- Created readthedocs documentation.
- Fixed bug where inbox and outbox folders were not created on saving partners and orgs.
- Fixed bug where MDN search was failing due to orphaned MDNs.

4.5.19 0.1.1 - 2015-09-04

- Increased the max length of MODE_CHOICES model field.
- Detect Signature Algorithm from the MIME message for outbound messages.

4.5.20 0.1 - 2015-04-29

- Initial release.