ICOS Stations and MSA Mailing Lists

Contents

Intro

ICOS-CP is running a Mailman3 installation to provide mailing lists to the ICOS community. This document explains the workings of a python script which creates and updates ICOS Station & Member State (MSA) email lists.

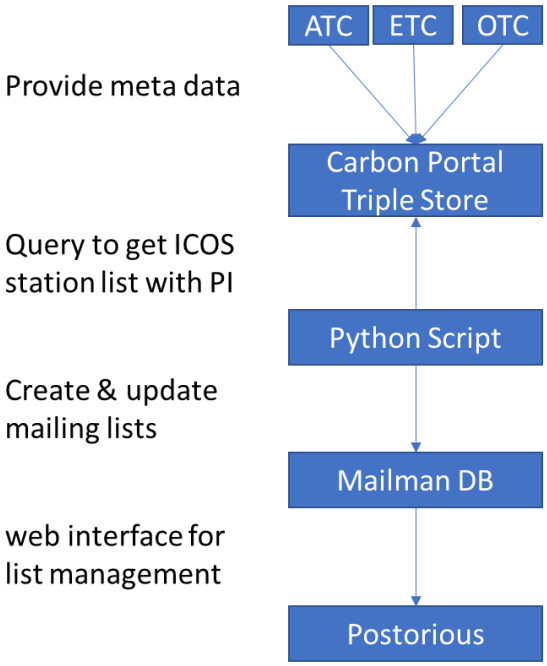
The general idea is that this script

- runs continously (Cronjob? Once a day?)
- queries the ICOS-CP SPARQL endpoint for a station list including PI information
- creates a mailing list for each station
- add PI as owner to the station list
- creates three overarching (umbrella) lists for MSA-atmosphere, MSA-ecosystem, MSA-Ocean add the station email list to the corresponding umbrella

File structure an their meaning:

cpmailman.py	This is the main script which should be invoked by the cron job. There is only one variable inside you may need to adjust, which points to the config.ini file
config/config.ini	This is the main configuration file, including the definition to the logfile name and where the umbrella list definition is found (list.ini)
config/list.ini	Definition of the three “umbrella” list
cphelpers.py	Common functions called from the cpmailman.py script. The SPARQL query and connection are defined here.
cpmailman.log	The name of this file is defined in config.ini. All log entries are stored in here

Simplified Data Flow



Good to know:

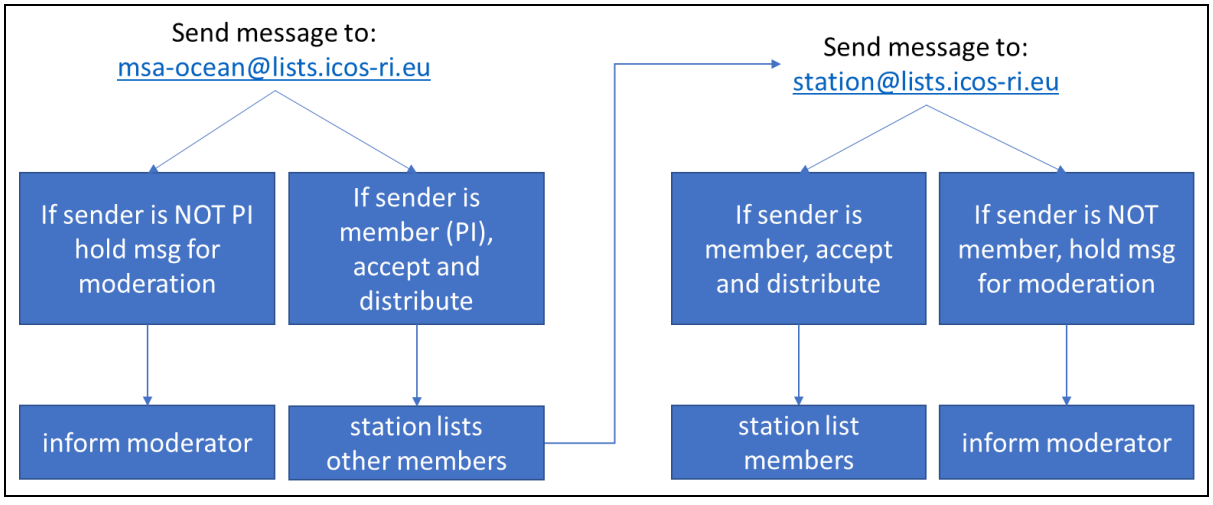
- Do NOT use quotes for strings in the config files. Parsing the config file automatically imports all values as strings, hence we would end up with qutoes inside the string.
- No message (email) will be sent while creating the list, owners and members.
- The SPARQL query used is a copy of “provisional stations PI’s” from the website
- The station list name is the “station id” + domain. All whitespaces are removed, and the lists are in lower case. For example the Ocean station with ID “Thornton Buoy” will be thorntonbuoy@lists.icos-ri.eu
- The script does NOT delete anything. If a list is not existing it will be created. If the PI is not owner, it will be added to the owners list, it does not overwrite or remove the existing one. The same goes for moderators and members.
- The example .ini files contain explanations about each section. Just in case, they are reproduced below in this document.
- The section name for the umbrella list is used (hard coded) to divide the stations into their umbrella list. Don’t change it.

The result:

Every ICOS station (the result of the SPARQL query), get’s an email list, with the PI as owner and moderator. The PI can manage the station list through the webinterface (https://lists.icos-ri.eu). If the PI adds more people to the station list they will receive all emails sent to the msa-list. Additional members are not automatically added as members to the msa-list.

PI’s are not automatically created as users in the system. If they want to actively manage the station list, they need to sign up with the email address provided from the thematic centre (The PI’s email address in the meta data store).

Each section in the list.ini file creates an umbrella lists, containing all station lists as members, and all PI’s as member. Hence PI’s are allowed to send messages to the msa list. Vice versa, the umbrella list is added as member to the station list to allow sending messages.



The automation process stops here. It is the PI's responsibility and decision, to add /remove people from the station list. While creating the msa lists, a owner and moderator is defined in the list.ini file. It is their responsibility to add more members (if desired), like people from the Thematic Centre, Carbon Portal or the Head Office.

Files

Config.ini

```
# Configurgartion for the carbon portal mailman interaction
# please do not use quotation marks, all values will be strings
# after importing.


[mm_settings]
# all the setting to connect to the mailman api

url    = https://lists.icos-ri.eu/rest/3.0/
user   = restadmin
pass   = "bother Andre"
domain = lists.icos-ri.eu
# ------------------------------------------------------------

[cp_test]

# if test ist "TRUE" the PI email and name will be replaced with
# email and name. Just to make really sure, that no unwanted emails
# like welcome message, or mails about messages hold for moderation
# are by mistake sent to the stations PI

test    = True
email   = claudio.donofrio@nateko.lu.se
name    = claudio donofrio
# ------------------------------------------------------------


[cp_logger]
# where and how much to log
name     = cplogger
fileName = cpmailman.log

# set log level         {CRITICAL | ERROR | WARNING | INFO | DEBUG }
# with numerical value: {50 | 40 | 30 | 20 | 10 }
# this means that "value" and above are logged
# example: 30 -> log warning, error, critical

level    = 20
# ------------------------------------------------------------

[cp_listconfig]
# which .ini file contains the umbrella list definition?

# fileName = list.ini
fileName = test.ini
# ------------------------------------------------------------
```

```
# this is an additional owner, added to all the
# station and umbrella lists

[cp_admin]
owner = claudio.donofrio@nateko.lu.se
```

list.ini

```
# define the umbrella list for each thematic centre
# theses lists will contain all corresponding icos
# stations. The "link" between umbrella list and stations
# is the theme. Hence the "section name" needs to correspond
# to the theme coming from the sparql query.

[AS]
theme                   = atmosphere
name                    = cd_test_atmosphere

display_name            = cd_test_atmosphere
subject_prefix          = [msa_test]
owner                   = Claudio.Donofrio@nateko.lu.se
ownerDisplay            = Claudio Donofrio
moderator               = claudio.donofrio@nateko.lu.se
moderatorDisplay        = Claudio Donofrio
description             = Automatic Member State Assembly List
info                    = This email list is automatically created.
                          All the ICOS stations for the theme are added
                          automatically with the PI set as moderator.
advertised              = False
subscription_policy     = moderate
send_welcome_message    = False

# possible entries are: { public | private | never }
archive_policy          = never



[ES]
theme                   = ecosystem
name                    = cd_test_ecosystem

display_name            = cd_test_ecosystem
subject_prefix          = [msa_test]
owner                   = Claudio.Donofrio@nateko.lu.se
ownerDisplay            = Claudio Donofrio
moderator               = claudio.donofrio@nateko.lu.se
moderatorDisplay        = Claudio Donofrio
description             = Automatic Member State Assembly List
info                    = This email list is automatically created.
                          All the ICOS stations for the theme are added
                          automatically with the PI set as moderator.
advertised              = False
subscription_policy     = moderate
send_welcome_message    = False

# possible entries are: { public | private | never }
archive_policy          = never

[OS]
theme                   = ocean
name                    = cd_test_ocean

display_name            = cd_test_ocean
subject_prefix          = [msa_test]
owner                   = Claudio.Donofrio@nateko.lu.se
ownerDisplay            = Claudio Donofrio
moderator               = claudio.donofrio@nateko.lu.se
moderatorDisplay        = Claudio Donofrio
description             = Automatic Member State Assembly List
info                    = This email list is automatically created.
                          All the ICOS stations for the theme are added
```

```
                          automatically with the PI set as moderator.
advertised              = False
subscription_policy     = moderate
send_welcome_message    = False

# possible entries are: { public | private | never }
archive_policy          = never
```

cpmailman.py

```
# -*- coding: utf-8 -*-
"""
    Dynamically create and update emailing lists in mailman "lists.icos-cp.eu".
    The SPARQL endpoint is queried to get a list of ICOS Stations and the
    corresponding Principal Investigator with email address.
    For each "theme" (ecosystem, ocean, atmosphere)
    an email list is created/updated.
    Each station will have an "own" list where the PI is moderator.
    Most of the variables are defined in cpmailman.ini
"""


# import necessary modules
import helpers
import logging
import configparser
from mailmanclient import Client
from mailmanclient import MailmanConnectionError


# the configFile is the only variable you need
# to change. Everything else should be defined in .ini file

configFile = 'config.ini'

# ------------------------------------------------------------
# read the main configuration file

cp_config = configparser.ConfigParser()
cp_config.clear()
cp_config.read(configFile)


# ------------------------------------------------------------
# create a simple python logger
log = logging.getLogger(cp_config['cp_logger']['name'])
log.handlers.clear()

handler = logging.FileHandler(cp_config['cp_logger']['fileName'])
formatter = logging.Formatter(
        '%(asctime)s %(name)-12s %(levelname)-8s %(message)s')
handler.setFormatter(formatter)
log.addHandler(handler)
log.propagate = False
# ------------------------------------------------------------
# set log level CRITICAL | ERROR | WARNING | INFO | DEBUG )
# with numerical value: {50 | 40 | 30 | 20 | 10 }
log.setLevel(int(cp_config['cp_logger']['level']))
log.debug('logging started with level ' + cp_config['cp_logger']['level'])

####################
# start main prog #
####################

log.info('starting ICOS mailman sync')


# ------------------------------------------------------------
# read the list configuration
cp_lists = configparser.ConfigParser()
cp_lists.clear()
cp_lists.read(cp_config['cp_listconfig']['fileName'])
if not cp_lists.sections():
    log.warning('no lists to process')
```

```
    log.debug('read configuration  from file: ' +
cp_config['cp_listconfig']['fileName'])
    raise SystemExit()
else:
    log.debug('umbrella lists to process: ' + str(cp_lists.sections()))
# ------------------------------------------------------------

# connect to the mailman list server
client = Client(cp_config['mm_settings']['url'],
                cp_config['mm_settings']['user'],
                cp_config['mm_settings']['pass'])

try:
    log.debug(client.system)
except (MailmanConnectionError):
    log.critical(MailmanConnectionError)
    raise SystemExit(0)

mm_domain = client.get_domain(cp_config['mm_settings']['domain'])
mm_lists = mm_domain.get_lists()
log.debug(str(mm_domain))
log.debug('lists: ' + str(mm_lists))
log.info('connected to : ' + str(client))

# ------------------------------------------------------------
# loop through the umbrella lists defined in the .ini file
# create the list if it does not exist
# update the lists with the information from the .ini file

for list in cp_lists.sections():

    listname = cp_lists[list]['name'].lower()

    # create the list if it does not exist
    if not listname in str(mm_lists):
        l = mm_domain.create_list(listname)
        log.info('new list created: ' + listname)

    # read the list
    lst = client.get_list(listname + '@' +
cp_config['mm_settings']['domain'].lower())

    # get the settings from ini file and update the list configuration
    settings = lst.settings
    settings['display_name'] = cp_lists[list]['display_name']
    settings['subject_prefix'] = cp_lists[list]['subject_prefix']
    settings['description'] = cp_lists[list]['description']
    settings['info'] = cp_lists[list]['info']
    settings['advertised'] = cp_lists[list]['advertised']
    settings['archive_policy'] = cp_lists[list]['archive_policy']

    # make sure the list is open, to accept members without sending an email
    settings['subscription_policy'] = 'open'
    settings['send_welcome_message'] = cp_lists[list]['send_welcome_message']
    settings.save()

    log.info('settings updated for ' + settings['fqdn_listname'] )

    # add the carbon portal admin from config.ini
    if not lst.is_owner(cp_config['cp_admin']['owner']):
        lst.add_owner(cp_config['cp_admin']['owner'])

    # add owner and moderator from list.ini
    if not lst.is_owner(cp_lists[list]['owner']):
        lst.add_owner(cp_lists[list]['owner'],
                      display_name=cp_lists[list]['ownerDisplay'])

    if not lst.is_moderator(cp_lists[list]['moderator']):
```

```
        lst.add_moderator(cp_lists[list]['moderator'],
                          display_name=cp_lists[list]['moderatorDisplay'])

    # add owner and moderator as membmer, otherwise you will not get any emails
    if not lst.is_member(cp_lists[list]['owner']):
        lst.subscribe(cp_lists[list]['owner'],
                      display_name=cp_lists[list]['ownerDisplay'],
                      pre_verified=True,
                      pre_confirmed=True)

    if not lst.is_member(cp_lists[list]['moderator']):
        lst.subscribe(cp_lists[list]['moderator'],
                      display_name=cp_lists[list]['moderatorDisplay'],
                      pre_verified=True,
                      pre_confirmed=True)
    log.debug(lst)
    log.debug('owners: ' + str(lst.owners))
    log.debug('moderators: ' + str(lst.moderators))
    log.debug('members: ' + str(lst.members))

    # now we can set the subscription policy as defined in the ini file
    settings['subscription_policy'] = cp_lists[list]['subscription_policy']
    settings.save()

    log.info('finished to set up umbrella list: ' + cp_lists[list]['name'])


# ------------------------------------------------------------
# query the sparql endpoint for a list of stations with PI's and email
cpresult = helpers.sparql(helpers.stationQString)

# the sparql query function returns a tuple
# with the first entry 'false' if something went wrong
if not cpresult[0]:
    log.warning(cpresult)
    log.warning('sparql query not succesful')
    raise SystemExit(0)
else:
    log.info('sparql list contains ' + str(len(cpresult[1])) + ' entries')

# ------------------------------------------------------------
# Loop through all the stations and create a list

# re-read all the lists from the mailman server
mm_lists = mm_domain.get_lists()

for icos_station in cpresult[1]:
    station = dict(zip(cpresult[0], icos_station))
    listname = station['stationId'].lower() + '@' +
cp_config['mm_settings']['domain']

    # make sure there are not whitespace, maybe we need a good regex
    listname = helpers.cleanStr(listname)
    name  = station['firstName'] + ' ' + station['lastName']
    email = station['email']


    log.info('processing ' + listname)
    ################ ONLY TO TEST
    name  = 'Claudio Donofrio'
    email = 'claudio.donofrio@nateko.lu.se'
    ###############  TEST FINISHED

    # create a new list, if it does not exist
    if not listname in str(mm_lists):
        try:
            mm_domain.create_list(helpers.cleanStr(station['stationId']))
            log.info('new list created: ' + listname)
```

```
        except:
            log.warning("problem to create: " + listname)

    # read the list
    lst = client.get_list(listname)

    settings = lst.settings
    settings['display_name'] = station['stationName']
    settings['subject_prefix'] = "[ICOS_station_" + station['stationId'] + "]"
    settings['description'] = 'Automatic created ICOS station list'
    settings['advertised'] = False
    settings['archive_policy'] = 'never'
    # make sure the list is open, to auto accept new member
    settings['subscription_policy'] = 'open'
    settings['send_welcome_message'] = False
    settings.save()

    log.debug('list settings updated for station ' + settings['fqdn_listname'] )

    # now add owner and moderator and subscribe as member

    if not lst.is_owner(email):
        lst.add_owner(email, display_name=name)

    if not lst.is_owner(cp_config['cp_admin']['owner']):
        lst.add_owner(cp_config['cp_admin']['owner'])

    if not lst.is_moderator(email):
        lst.add_moderator(email, display_name=name)

    if not lst.is_member(email):
        lst.subscribe(email, display_name=name,
                      pre_verified=True,
                      pre_confirmed=True)


    # find the appropriate umbrella list and settings

    ulist = ''
    usetting = ''
    try:
        umbrella = cp_lists[station['stationTheme']]['name'] \
                   + '@' + cp_config['mm_settings']['domain']
        ulist = client.get_list(helpers.cleanStr(umbrella))
        usetting= ulist.settings
        usetting['subscription_policy'] = 'open'
        usetting.save()

        try:
            # add the station list as member to the umbrella
            if not ulist.is_member(settings['fqdn_listname']):
                ulist.subscribe(settings['fqdn_listname'],
                                pre_verified=True, pre_confirmed=True)

        except:
            log.error('adding ' + settings['fqdn_listname']
            + ' to ' + usetting['fqdn_listname'])

        try:
            # add the PI from the station as member to the umbrella
            if not ulist.is_member(email):
                ulist.subscribe(email, display_name=name,
                                pre_verified=True, pre_confirmed=True)

        except:
            log.error('adding PI ' + settings['fqdn_listname']
                + ' to ' + usetting['fqdn_listname'])
```

```
    except:
        log.error('umbrella not found for: '
                  + station['stationTheme'] + ' - '
                  + listname)


    settings['subscription_policy'] = 'moderate'
    settings['send_welcome_message'] = True
    settings.save()

    log.debug(lst)
    log.debug('owner: ' + str(lst.owners))
    log.debug('moderators: ' + str(lst.moderators))
    log.debug('members: ' + str(lst.members))



"""
    This is to test and debug.
    All the created mailings list, can automatically be removed
    by setting deletet = True
    delete = True
"""

delete = True
if delete:
    for icos_station in cpresult[1]:
        station = dict(zip(cpresult[0], icos_station))
        listname = station['stationId'].lower() + '@' +
cp_config['mm_settings']['domain']
        log.info('delete ' + listname)
        # make sure there are not whitespace, maybe we need a good regex
        listname = helpers.cleanStr(listname)
        lst = client.get_list(listname)
        lst.delete()
        log.debug('delete '  + str(listname.lower()))

    for list in cp_lists.sections():
        listname = cp_lists[list]['name'] + '@' +
cp_config['mm_settings']['domain']
        lst = client.get_list(listname.lower())
        log.info('delete ' + lst)
        lst.delete()
        log.debug('delete '  + str(listname))


log.info("ICOS mailman sync finished")
log.info("-----------------------------------------")

# -- EOF --- ICOS mailman sync finished
```

helpers.py

```
             # -*- coding: utf-8 -*-

"""This file contains common functions, tools and utilities"""
__version__= "0.1.0"


# load necessary modules
import requests

# create helper functions
#---------------------------------------------------------------------

def is_number(num):
    """
    check if param can be converted to a float
    param:
    return: bool [True|False]
    """
    try:
        float(num)
        return True
    except ValueError:
        return False

#---------------------------------------------------------------------

def debugPrint(dbg, msg):
    """
    prints debug information to the console if param1 is "True"
    param1: bool (True|False)
    param2: String or object which is printablemessage
    """
    if(dbg):
        print(msg)

#---------------------------------------------------------------------

def checklib(module):
    """ load a list of modoules if available, otherwise throw exception """
    import imp
    for mod in module:
        try:
            imp.find_module(mod)
            ret = 1
        except ImportError as imperror:
            print(imperror)
            ret = 0
    return ret

#---------------------------------------------------------------------
def stationQString(*args):
    """
        Define SPARQL query to get a list of ICOS stations with PI and email
    """
    query = """
            prefix st: <http://meta.icos-cp.eu/ontologies/stationentry/>
            select distinct ?stationTheme ?stationId ?stationName ?firstName
?lastName ?email
            from <http://meta.icos-cp.eu/resources/stationentry/>
            where{
                ?s st:hasShortName ?stationId .
                ?s st:hasLongName ?stationName .
                ?s st:hasPi ?pi .
                ?pi st:hasFirstName ?firstName .
```

```
                ?pi st:hasLastName ?lastName .
                ?pi st:hasEmail ?email .
                ?s a ?stationClass .
                BIND (replace(str(?stationClass), "http://meta.icos-
cp.eu/ontologies/stationentry/", "") AS ?stationTheme )
            }

        """

    return query
#-------------------------------------------------------------------------------
def sparql(queryString):
    """
        This functions queries the ICOS sparql endpoint
        with param1, queryString. By default the returned object
        is a python tuble with two arrays.
        The first array contains the column names and the second array
        contains the "result" (binding) in each row.
    """

    # Query the ICOS SPARQL endpoint for a station list
    # output is an object "data" containing the results in JSON
    #-------------------------------------------------------------------------


    url = 'https://meta.icos-cp.eu/sparql'
    r = requests.get(url, params={
        'format': 'json',
        'query': queryString()})

    if not r.ok:
        return r.ok, r.reason

    data = r.json()
    #-------------------------------------------------------------------------

    # convert the the result into two arrays
    # cols = column names
    # datatable = results

    cols = data['head']['vars']
    datatable = []

    for row in data['results']['bindings']:
        item = []
        for c in cols:
            item.append(row.get(c, {}).get('value'))

        datatable.append(item)

    return cols, datatable
#-------------------------------------------------------------------------------
def cleanStr(txt):

    #remove whitespace
    txt = txt.replace(' ','')
        # make sure everythig is lowercase
    txt = txt.lower()

    return txt
#-------------------------------------------------------------------------------
```