Factory

MirosRabbitManualNetwork

make_amqp_url(ip_address)

routing_key
exchange_name
dict

hosts
live_hosts
dead_hosts
live_amqp_urls
dead_amqp_urls

file_name
file_path
manual_file_chart

ManNetChart

Event Processor

```
ManNetChart.DEFAULT_JSON = '''
{
  "hosts": [
  ]
}
'''
```

read_and_evaluate_network_details

```
entry /
  if not hasattr(chart, 'manual_file_chart'):
    chart.file_path = '.miros_rabbimq_hosts.json'
    chart.file_name = os.path.basename(chart.file_path)
    chart.manual_file_chart = CacheFileChart(
      file_path=chart.file_path, default_json=ManNetChart.DEFAULT_JSON)
  chart.subscribe(Event(signal=signals.CACHE))
  chart.subscribe(Event(signal=signals.AMQP_CONSUMER_CHECK))
  chart.publish(Event(signals.CACHE_FILE_READ))

network_evaluated /
  payload = ConnectionDiscoveryPayload(
    ip_addresses=chart.live_hosts,
    amqp_urls=chart.live_amqp_urls,
    from=chart.name)
  chart.publish(
    Event(signal=signals.CONNECTION_DISCOVERY, payload=payload)
```

```
CacheReadPayload = \
  namedtuple('CacheReadPayload',
    ['dict', 'last_modified', 'created_at', 'expired', 'file_name'])
```

```
CACHE as e: \
  [e.payload.file_name == chart.file_name]
  chart.hosts = e.payload.dict['hosts']
```

```
ConnectionDiscoveryPayload = \
  namedtuple('ConnectionDiscoveryPayload',
    ['hosts', 'amqp_urls', 'dispatcher'])
```

evaluate_network

```
entry /
  chart.candidates = {}
  for host_address in chart.hosts:
    chart.candidates[host_address] = \
      RecceNode(searched=False,
        result=False
        scout=RabbitConsumerScoutChart(
          host_address, chart.routing_key, chart.exchange_name))
```

each scout will produce
an AMQP_CONSUMER_CHECK event
and will be destroyed after use

```
AMQP_CONSUMER_CHECK as e with payload /
  h, result = e.payload.ip_address, e.payload.result
  is_one_of_my_hosts = h in chart.hosts
  is_my_routing_key = e.payload.routing_key is chart.routing_key
  is_my_exchange_name = e.payload.exchange_name is chart.exchange_name

  if is_one_of_my_hosts and is_my_routing_key and is_my_exchange_name:
    chart.candidates[h] = RecceNode(searched=True, result=result, scout=None)
    if result:
      chart.live_hosts.append(h)
      chart.live_amqp_urls.append(chart.make_amqp_url(h))
    else
      chart.dead_hosts.append(h)
      chart.dead_amqp_urls.append(h)
    search_complete = all([node.searched for node in chart.candidates.values()])

    if search_complete:
      chart.post_fifo(Event(signals.network_evaluated))

CACHE / {}
```

```
RecceNode
  namedtuple(
    'RecceNode', ['searched', 'result', 'scout'])
```

EVALUATE_HOSTS_FILE

CONNECTION_DISCOVERY(<ConnectionDiscoveryPayload>)