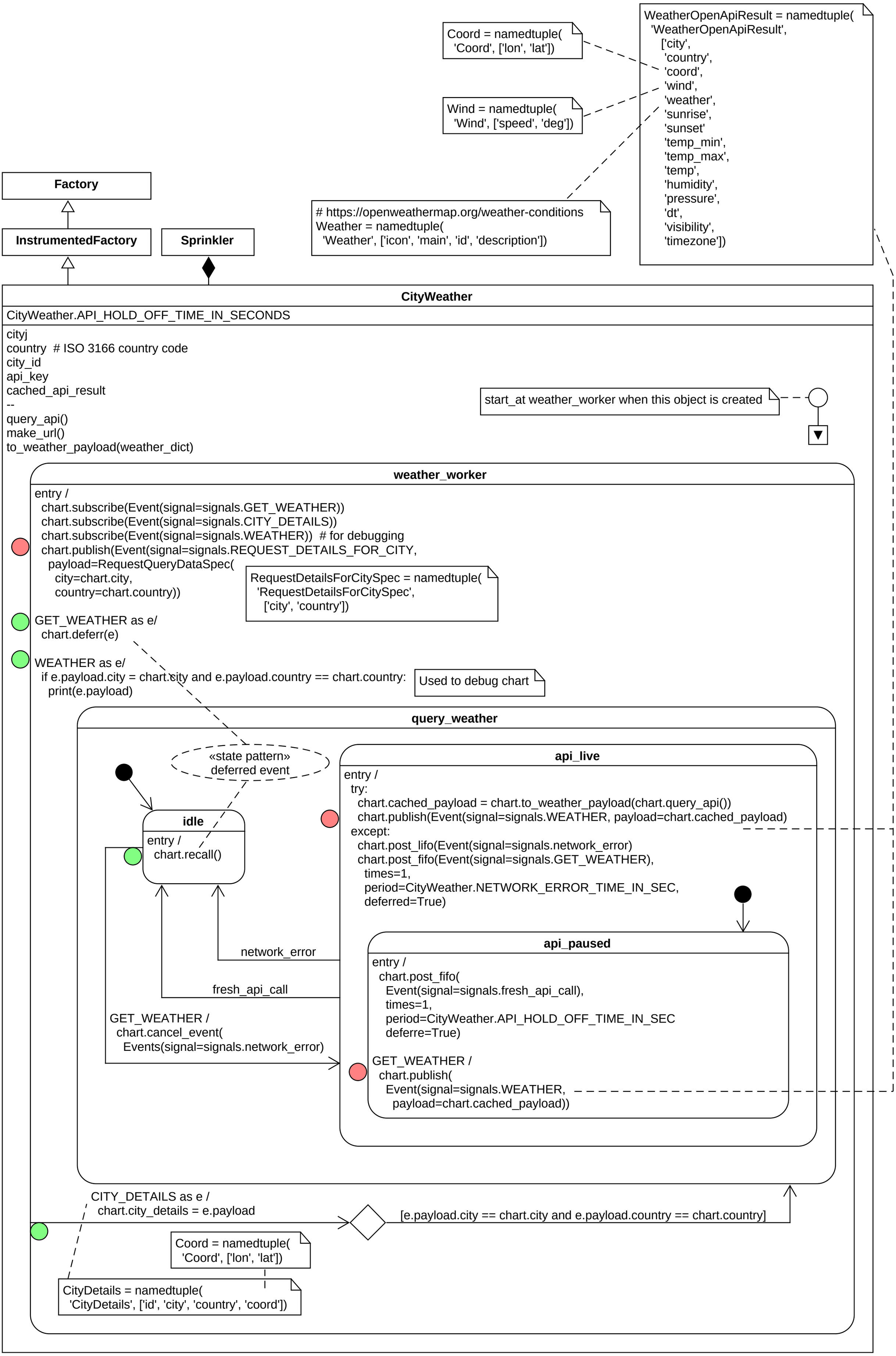
Coord = namedtuple(
'Coord', ['lon', 'lat'])
Wind = namedtuple(
'Wind', ['speed', 'deg'])
WeatherOpenApiResult = namedtuple(
'WeatherOpenApiResult',
['city',
'country',
'coord',
'wind',
'weather',
'sunrise',
'sunset'
'temp_min',
'temp_max',
'temp',
'humidity',
'pressure',
'dt',
'visibility',
'timezone'])
# https://openweathermap.org/weather-conditions
Weather = namedtuple(
'Weather', ['icon', 'main', 'id', 'description'])
Factory
InstrumentedFactory
Sprinkler
CityWeather
CityWeather.API_HOLD_OFF_TIME_IN_SECONDS
cityj
country # ISO 3166 country code
city_id
api_key
cached_api_result
--
query_api()
make_url()
to_weather_payload(weather_dict)
start_at weather_worker when this object is created
weather_worker
entry /
chart.subscribe(Event(signal=signals.GET_WEATHER))
chart.subscribe(Event(signal=signals.CITY_DETAILS))
chart.subscribe(Event(signal=signals.WEATHER)) # for debugging
chart.publish(Event(signal=signals.REQUEST_DETAILS_FOR_CITY,
payload=RequestQueryDataSpec(
city=chart.city,
country=chart.country))
RequestDetailsForCitySpec = namedtuple(
'RequestDetailsForCitySpec',
['city', 'country'])
GET_WEATHER as e/
chart.deferr(e)
WEATHER as e/
if e.payload.city = chart.city and e.payload.country == chart.country:
print(e.payload)
Used to debug chart
query_weather
«state pattern»
deferred event
idle
entry /
chart.recall()
network_error
fresh_api_call
GET_WEATHER /
chart.cancel_event(
Events(signal=signals.network_error)
api_live
entry /
try:
chart.cached_payload = chart.to_weather_payload(chart.query_api())
chart.publish(Event(signal=signals.WEATHER, payload=chart.cached_payload)
except:
chart.post_lifo(Event(signal=signals.network_error)
chart.post_fifo(Event(signal=signals.GET_WEATHER),
times=1,
period=CityWeather.NETWORK_ERROR_TIME_IN_SEC,
deferred=True)
api_paused
entry /
chart.post_fifo(
Event(signal=signals.fresh_api_call),
times=1,
period=CityWeather.API_HOLD_OFF_TIME_IN_SEC
deferre=True)
GET_WEATHER /
chart.publish(
Event(signal=signals.WEATHER,
payload=chart.cached_payload))
CITY_DETAILS as e /
chart.city_details = e.payload
[e.payload.city == chart.city and e.payload.country == chart.country]
Coord = namedtuple(
'Coord', ['lon', 'lat'])
CityDetails = namedtuple(
'CityDetails', ['id', 'city', 'country', 'coord'])