version v0.5 at July 13th 2016 - contract ABI is not published yet, but you can start already - by reading the basics.

Smart Contract: `WeatherForecastAtLocation`

```
Novel smart contract solution for Ethereum based platforms, here: SOIL.
On-chain delivery of weather forecasts, for any mapped location on Earth.

User-paid request selects location - an oracle answers to that event.
Then 1 nowcast, and 4 forecasts can be read by anyone, for 4 days.
Auto-updating function is planned, for continuous supply of fresh data.

System is already functional (soon), please start testing it.
```

A proof of concept for reliable data delivery into a blockchain, so that *other smart contracts* can utilize the information. The era of "smart contracts" has just begun, and we do not grasp yet what will be the best use cases for this nascent technology - so let's just start with something. Still, you might ask:

Why Weather?

The natural environment is at the basis of many ecological questions, and good weather forecasts have been helping us a lot, for various decision making. During this century they will become even more important when the centuries-old wisdoms of long-term climate insights are loosing their reliability. The transitional future might look like: Less climate, more weather.

Consequences of weather are felt not much anymore by city-dwellers - but much by people whose lives depend on rain, wind, clouds, temperature. To harvest one day too late, or one week to early - can decide about the survival of a farm, family, village. To put a new roof on a house - is easier and safer during calm weather. A festivity - can sometimes be chosen to, and prepared to be take place outdoors or indoors. Large scale technological systems like photovoltaic, and wind energy farms - need to be balanced well with other means of energy production and storage - and their valuable outputs are direct functions of (rain, wind, clouds, temperature).

Weather forecasts are inherently difficult. Not only literally - because they are about the future - but because the weather system is highly complex. While it can be very stable over days or weeks (then "the best forecast for tomorrow is today's weather"), weather can also change dramatically, and quickly. It has complex components, and frequent temporal phases of: non-linearity. Here non-linearity describes that small input changes are able to quickly evolve into large outcome differences (coined "butterfly effect", in a 1972 talk) - studied in a field called chaos theory, quite a misleading term actually.

Long-term weather forecasts are still near to impossible, and might always be. For the short term, however, throughout our lifetimes we have seen immense improvements in the reliability of weather forecasts - in short: For many weather domains, a range of 3 days can nowadays be predicted pretty well.

Ecologically oriented "smart contracts" - why weather?

This project is not providing a specific solution for any ecological question, but it lays the groundwork for many processes, and structures to be modeled into blockchain-based smart contracts.

Some examples that - are all unsolved as smart contracts but - could benefit from weather forecasts as input into their automated decision process:

- insurance against loss of harvest through rain, or lack thereof (*)
- e.g. hay needs to be dry; cotton, and wheat suffer from long periods of dryness
- commodity trading of agricultural produce; related financial contracts and derivatives (*)
- predict rising peach prices already before a spring freeze during the blooming season (*)
- automated watering of plants could be stopped many hours before it rains anyways
- emptying (e.g. a water reservoir for) energy storage of wind energy, depending on wind forecasts for the next days
- last minute plane ticket sales in storm-risky times (which might lead to closed airports) (*)

- utility companies are estimating likely demands in the next days (aircon, heating)
- supermarkets could stock different products depending on next weekend's weather
- swim suit or sweater - your "smart wardrobe" might tell you which clothes to wear for today
- automated delivery drones need ways to avoid starting during storm

Please be creative, and make suggestions for that list. Keep in mind that the short term (0-3 days) version of the same question is focused. About the (*) read the next chapter:

How to make blockchains aware of the real world, and: a warning!

While some of the above questions have -and also computational- solutions that are depending on web based weather forecasts, *blockchains can only see one thing: their own past*. As by themselves blockchains are completely unaware of the outside world, the simple contract "check weather forecast, and in case of frost tomorrow night: sell 1/3 of my stocks in peach growing companies" ... is still impossible, as the needed input data is just not there. Due to the nature of the question, the first solutions for "stocks in peach companies" are already working well as blockchain implementations (decentralized asset exchanges allow to find market prices, dividends can be paid, voting rights executed in relation to the ratio of held stocks, etc.), but questions related to the state of the outside world need new approaches, and third party solutions.

Third parties introduce the risk of trust. Especially if the information itself can be used to directly manipulate a trade, there is no reason to believe a single actor when he is telling you e.g. a price of a thing - because he might want to buy, or sell it; and lie about the truth. For example, if a blockchained asset is traded as a proxy for the current USD-EUR forex price, trustless mechanisms need to be put in place to approximate the "outside world's truth" about the "real" USD-EUR price. One of the suggestions is called SchellingCoin, where votes about a truth from many actors are rewarded for their (intersubjective) truthfulness; and as there is an incentive, you can hope that they collectively converge to something that resembles a "truth". Among other conditions (and pitfalls), it needs enough participants who are interested in telling you about that truth, and willing to risk their own funds in the process.

Here, as a first approximation, I suggest a different approach: Let us simply trust the actors who are reporting the weather into the blockchain, as a service they are getting directly paid for. For many cases such overly trust will indeed be without much risk involved. Other use cases (examples marked with a (*) above) should probably still be avoided or treated with scepticism - until a fully trustless weather forecasting solution has been found, and implemented (And note: To later then change a data source for your innovative smart contracts that is weather depending might not be difficult so this current non-trustless approach here can provide a kickstarter for a whole new field. And make the development of a really trustless weather forecast service even probable.

Yes, financial applications were historically the first ones to be reliably implemented on blockchains - because they benefit so much from the provided trustless paradigm. But a "world computer" can do much more for us than replacing outdated implementations of what is essentially ... only money.

Oracles

Intermediaries that transfer outside world knowledge into the blockchain are called 'oracles'. It reminds of the famous Pythia, the Oracle of Delphi, which 3 millennia ago provided answers for otherwise unanswerable questions, probably enabled by hallucinogens, and - in return for cash, a 'sacrificial gift'. The 4 step process described on the Wikipedia page is surprisingly comparable to this project:

- 1 Journey to Delphi = reading these texts, installing software, learning the underlying technology.
- 2 Preparation of the Supplicant = loading ABI of smart contract, helper tools; identifying locationSpecifier.
- 3 Visit to the Oracle = sending locationSpecifier request to the smart contract, and some money.
- 4 Return Home = getting weather forecasts for that locationSpecifier, for the following 4 days.

I suggest once you have finished your hands-on first experiments with this smart contract, that you then return to that wikipedia page about Pythia - for entertainment, and cultivation :-) And by the way: *Pythia* - part of the work of the past weeks was to create new *Python* tools that'll help you much.

Oracle contract: "WeatherForecastAtLocation"

Our oracle loop listens to events that a **blockchain user is "topping up"** a specifier for the **location** the user is interested in. Then the oracle queries a **weather** forecasting service (with generous CC BY-SA 4.0 licensing conditions - openweathermap), and uploads the data into a **smart contract storage**. Now the (any!) user can come back, and get this **forecast** data (for free). Until the forecast is outdated.

The in-chain answer is available usually 2-5 blocks after the in-chain question. Approximately 1 minute. (A consequence of the low 'clock rate of the world computer', of ~0.05 Hz.)

For this contract version, 5 points in time are stored: **Now, tomorrow, in 2 and 3 days, and 96 hours from now**. A restrictive choice to not overload the blockchain unnecessarily with data that is probably not even needed. Extensions can be discussed: To return 2, or 4 values per 24 hours; or to always store the weather forecast *at noon* local time (to allow for location comparison). Or to provide a "one-shot" forecast for one certain timestamp only, but then more precise in terms of time.

For this first iteration, we are starting with a simple system, to collect experiences to then improve upon (and in case you need local time *noon* forecast it can already be solved by asking exactly at noon - because then the following 4 forecasts will be noon time too).

Any query **after the last timestamp** (i.e. the "far future") results in an error message. A query **before the first timestamp** (i.e. the past) simply returns the first timestamp's data- so an easy way to ask about the "now" is to ask about the **forecast at time 0**.

Nowcasting

Weather forecasting -even though done in impressively huge computer simulations- is unreliable. Reporting the current weather is incomparably easier. And still it provides valuable information that can be very important for dependent processes (e.g. "shall the delivery drone start now, or is it raining?"). So it was decided to provide two types of information - weather forecast, and current weather. If the queried timestamp is within the first 11:59 hours, the currently measured weather is returned; for the hours 12:00 to 35:59 after the now, the first forecast will be reported back.

Money talks

A possible future for this system is *one* central smart contract that gets fed with weather data by *many* oracles. The data structures & authorization functions are implemented already. The **oracle/s** get immediately rewarded for each upload, via sending them money directly, at a 50% rate. The other 50% are added to an internal balance that is rising with each newly uploaded weather timestamp, and that can be withdrawn by the **contract owner**. The owner can change that percentage in the running contract, new oracles can be added, old ones deleted, the minimum top up ('price per location top up') can be lowered or raised, etc. - at a later moment in time, all the non user_... functions will be documented too. The contract has got quite complex functionality, but let's **focus on the user-view** for now.

Money can get stuck, e.g. because a user is topping up a specifier for a location that simply does not exist - then this money can only be freed, when the owner is completely killing the contract (which happens often during development phase, but should almost never in production phase). Killing the contract means, all data is deleted, and all money is sent to the contract owner.

So please: Do not send a month's salary worth of coins to the contract, and always test a new location with a small amount first. And in general, very simple, all this is experimental, and not even in 'beta' yet: ***Consider the money gone, as soon as you send.*** (There will be an event emitted containing your address, but if no event recorder caught that, that info might also get lost.) **Important**: As the contract does *not* store *who* is sending money for a location, only the total location balance ... you will become one of the generous **sponsors** of that location, your money gets collectivized for a greater purpose - everyone can read one more location forecast then - but: **there are no refunds**!

During this experimental phase, 1 "top up" for 1 location costs you 5 SOIL (= for 1 USD you get 30 "top ups", i.e. 4 months worth of forecasts). For these 5 SOIL, the oracle gives everyone 1 nowcast, and 4 forecasts - covering 96 hours. The low costs are intended to allow for excessive testing, but still limit massive abuse, and blockchain spamming. Financing the development costs (of already well over 100 hours) seems unlikely, it would need >100,000 "top ups". So this whole project depends on winning awards and prizes, it will really benefit from your personal donation - and it needs to find **Angel Investors**. Please help.

You're next

Just start using the current solution, it is fully functional. Please start building your own smart contracts on top of it - only then we can together find out how to adapt it best to your real world use cases.

The exact specifics of when, how, which and how much data, and how much fees - are all open, and open for discussion. The platform is to be seen as a first proof of concept, not as the final solution.

Components

The general architecture is ready to use:

- Contract: user, oracle, owner - access-limited functions, depending on the role = sender's address.
- Oracle service: Event listener loop, weather query, upload into chain. Plus classical DB as persistant storage.
- Web Framework: Works with that classical DB data (admin platform ready, but no user view functions yet)
- Scripts & Config: ABI and JavaScript and Python, for easy communication with the contract.
- System setup & manuals: How to prepare your system. To use - and to develop smart contracts.

Which data exactly will then be communicated - is actually secondary. This proof of concept project has chosen *weather forecast data*, because of its wide applicability for environmental, and ecological questions. Which was the reason why to choose the SOIL blockchain to launch. Inspired by:

Hackathon

Parts of this project are a submission to the "SOILcoin dApp Hackathon", with the goal to stimulate dApp development about "RENEWABLE ENERGY, ENVIRONMENTAL TECHNOLOGY, PRECISION AGRICULTURE, COMPUTATIONAL SCIENTIFIC APPLICATIONS". That hackathon inspired the idea for this whole project, that has now become much larger than originally planned. At the current SOIL evaluation, even winning the first prize would only recover ~10% of the development costs - but it would help. Please cast your vote! There are no other submissions to the hackathon yet.

SOIL itself is in transition. And this project has grown too big already. To lower the risks for the author, and allow for a later crowdfunding (with more substantial return), keeping control of intellectual property makes sense. And smart contracts allow to publish working binary code without revealing the whole Solidity source code.

The suggested deliverables are:

- 1 smart contract blockchain service
  - deployed in compiled form on SOIL chain (address: tba)
  - ABI (smart contract interface definitions)
  - JS and Python tools for easy access
- 1 oracle service
  - oracle loop running on author's server (btc.hopto.org)
  - planned: user view frontend on top of traditional DB (which already gets populated by oracle loop)
- several tutorials: step by step manuals for all the user_... functions
  - in basic JavaScript - how to interact with the live smart contract, via the `gsoil attach` console
  - in elaborate Python - how to interact with the live smart contract, with many useful helper functions.
  - this text, and more.

Please give feedback, and VOTE FOR THIS PROJECT. (see SOIL thread)

Become an Angel:

The birth of innovative projects is motivated by curiosity, genius, and the identification of a real need. The continuation of a project is more likely if there is appropriate funding. Please send to

- [SOIL] 0x1077ac9ebf8e5c492b7174c3a3d264801044834e
- [BTC] 1J8X4PWeGEkKDsG3TbSZfzeAPdYmQMc5ut
- [ETH] 0x86c362fde6df8c5bcdb7091d97bd70bf06262bf8

Thank you. If the turnout is encouraging, a crowdfunding campaign is considered.