Importance

The covalent network begins with the creation of block specimen objects forming the core of the network's data objects specification. These objects are created with the aid of three main pieces of open-source software provided by Covalent for the network's decentralized stack.

1. Block Specimen Producer (BSP) - Validator operated & deployed
2. BSP-Agent - Validator operated & deployed
3. BSP-Proof-chain - Covalent pre-deployed

Functions

Go-ethereum (geth) is used as the underlying EVM based blockchain node that houses the BSP modifications. Once, go-ethereum is run with the correct configurations that enable this feature, block specimens are created as the node catches up historically from block 0 or even if it just starts syncing from the block number where it left last off after the geth process was stopped (with or without the patch). The validators have the ability to decide from which block they would like to start producing specimens.

Once the producer is correctly started, block specimens are created for every block that is executed and consequently synced with geth. These specimens are then sent off to a Redis streaming service (that the validators need to run on the same system as the geth process) at the particular topic key where the BSP-agent can listen for receiving these messages.

The BSP patch on geth makes sure to put as least as a possible effort on the geth node for the creation of these specimens while it can continue to operate in other geth specific modes. In this manner, the creation of messages is decoupled from the processing of the messages with an in-memory queue which handles the brokerage of block specimen messages sitting in between BSP and the BSP-Agent. The streaming service enables asynchronous processing of the messages with horizontal scalability for free.

Further, the BSP-Agent when run with the correct Redis configuration does the job of listening to the streamed messages on the same topic as the one provided with the BSP operation. Within the Agent operation, the flag configurations also allow us to specify how we would like to pack the incoming block specimen messages so that they can be stored in any storage service. The "how" refers to the given size (contents), encoding used (AVRO) and location (GCP) of the block specimens for storage.

Between reading each stream message and storing a block specimen object representing an evaluation (validating and proving) process the Agent performs a bunch of processing expensive operations including a single gas expending operation of writing the proof-of-creation

of block specimens to an(y) EVM based blockchain. The chain that the agent writes to is also specified during the initialization of the Agent.

In the order of processes, the Agent firstly makes sure that the message follows the format it expects to receive from a BSP - producer. If for example, it receives messages from another topic that does not conform to the expected schema, the application stops any further processing. Secondly, once the messages are verified to be in the correct format matching the block specimen schema, it proceeds to pack multiple stream messages into an array of block specimens within a single binary encoding using AVRO (a .avsc schema). The size of packing is specified at the initialization of the Agent. The Agent also ships with this schema it expects to receive from the Redis stream and pack into for storage.

The AVRO encoding for packing multiple block specimens is basically the process of segmenting the messages into batches and processing them for proving and storing them in those exact batches. Thirdly, the Agent keeps track of every stream message that's been verified and segmented into a batch and then encodes (compressing) them using this codec converting them into a single binary array of multiple specimens. Finally, after evaluating an entire segment the Agent has the capability to talk to a proof-chain contract deployed on the earlier specified blockchain node. It takes the binary specimen object that's ready for storage, does a SHA256 sum over its contents and calls a payable function on the BSP-Proof-chain contract that is pre-deployed to the corresponding node that it is talking to.

The Agent is able to obtain the correct credentials to make a transaction to the contract calling the proving function from its environment configuration, thereby allowing it to make statements regarding the measurable properties like exact size and contents of the block specimen segment. The agent effectively makes a provable (or dis-provable) commit regarding the work it's done so far on the block specimen segment object itself. Once this proof-of-creation transaction is mined, the Agent proceeds to upload the object it's already made the commit for to the specified object storage service location provided in initialization configuration.

The process of evaluating a message, segmenting multiple messages into a single AVRO encoding, making a statement regarding the work done by making a transaction on-chain and finally uploading the segment is an entirely atomic process. All of the above will either go through or none of them will. Anyone section failing in this chain at any point leads to the failure of the entire process within the Agent. In case of such failure, the exact reason is made as explicit as possible in the logs from where corrections can be applied during initialization. The messages in the redis stream are persisted in the queue, in case a failure in the processing chain happens thereby making sure that if at all - only fully evaluated, encoded and proved (processed) segments get uploaded to any storage service.

With this, the binary block specimen AVRO segment objects are stored with their filenames encoding the origin network id (in the case of mainnet will be 1), followed by the blocks for which the specimens are included in the given binary object. For example for the file “1-1-5”, block specimens 1 to 5 are included for chain id 1.