pyth2wormhole attestation wire format, version 3 rev. 0

Goals

This document outlines a purpose-built serialization format for Pyth2wormhole - the Pyth data bridge on Wormhole.

Non-goals

- Description of control flow guidelines
- Pyth2wormhole and Wormhole business logic design

Requirements

- **Maximum efficiency in storage and serialization/deserialization routines** - The message format must be deserialized in on-chain programs. On-chain computation is expensive, so an efficient format minimizes operational cost.
- **Forward compatibility** - we expect to target a diverse collection of smart contract environments. Because of their differences, updating all implementations of deserialization logic at once is a big ask. The design must allow for older logic to be as competent with newer format iterations as possible. This requirement is motivated by the likely addition of previously absent Pyth metadata items on an append-only basis.

Conventions

Byte-level format

For byte-for-byte serialization we commit to tightly packed bytes in big-endian byte order with two's complement representation for signed integers. Endianness choice serves efficiency - big endian is used by expensive chains such as Ethereum.

Versioning

Breaking changes

Any breaking change to this format that fails to leverage the forward compatibility measures deserves an increase of the `major_version` value in the header. Each new `major_version` specification must set `minor_version` to `0`. Deserialization routines are expected to refuse different `major_version` values from what is current at implementation time.

Non-breaking changes

Forward-compatible changes to this format must be accompanied with an increase of the `minor_version` field inside the header of a payload. Subsequent `minor_version` revisions should be accompanied by a change-based summary over their respective predecessors. Deserialization routines are free to reject payloads with `minor_version` values lower than what is current at implementation time.

Other considerations

- We advise production deployments wishing to bump `major_version` to implement a grace period where equivalent messages of both versions are available.

Nesting

Nested non-primitive data structures may not include a header if the context of surrounding data structure is deemed sufficient. The general aim should be towards more packing efficiency.

Variable-length data

Unless stated otherwise, all data structures have constant size.

Pyth2wormhole format message types

BatchPriceAttestation

Top-level pyth2wormhole message. Contains a variable number of uniform size `PriceAttestation` messages.

Note on `attestation_size`

`attestation_size` is a forward compatibility field that helps with adding more fields at the end of `PriceAttestation` messages inside a batch **without a `version` bump**. Newer serialization routines are free to append any extra price attestation metadata, as long as the alignment to the `attestation_size` they choose is preserved within the enclosing batch. Outdated deserialization routines must ignore the trailing bytes after consuming the format revision they know. Distinct batches are free to use different `attestation_size` values if necessary. To prevent “payload too short” parsing errors, the size value may never decrease for subsequent revisions of a given format version.

BatchPriceAttestation fields

Field Name	Type	Length in bytes	Description
header	Header		payload_id = 2, constant length equal to the size of a `Header`
n_attestations	uint16	2	Decides the number of `PriceAttestation` messages enclosed in this batch.
attestation_size	uint16	2	Describes the size in bytes of a single `PriceAttestation` in this batch
attestations	tightly packed PriceAttestation payloads		variable-length: length = n_attestations * attestation_size

Header

We use `P2WH` raw ASCII bytes as a magic byte string to mark a payload adhering to the format. We expect to increase the version field values as described in Versioning. All top-level messages must start with a header. `hdr_size` is constant within a single `minor_version`. New fields may be added after the last field on an append-only basis for forward compatibility.

Header fields

Field Name	Type	Length in bytes	Description
magic	bytes	4	Constant 4 ASCII bytes "P2WH", not terminated; sanity check that payload is not bogus
major_version	uint16	2	Constant value: 3; major version number of the format; bigger number means newer version
minor_version	uint16	2	Constant value: 0; minor version number; bigger means newer; newer minor versions must not break correctly implemented parsers within the same major version
hdr_size	uint16	2	Constant value: 1; size of remaining header fields in bytes
payload_id	uint8	1	Uniquely assigned number identifying the format of the following bytes

PriceAttestation

Used for communicating a single product's price state. For forward compatibility, new fields may be added to this struct on an append-only basis. Any messages containing a `PriceAttestation` are required to honor this property.

PriceAttestation fields

Field Name	Type	Length in bytes	Description
product_id	bytes32	32	solana account key of the product
price_id	bytes32	32	solana account key of the price; used for disambiguation between different prices on a single product
price	int64	8	PriceInfo `price` field
conf	uint64	8	PriceInfo `conf` field.
expo	int32	4	Price `expo` field; denotes price value exponent
ema_price	int64	8	Price `ema_price` (formerly `twap`) field
ema_conf	uint64	8	Price `ema_conf` (formerly `twac`) field
status	bytes	1	PriceInfo `status` field in u8 representation

Field Name	Type	Length in bytes	Description
num_publishers	uint32	4	PriceInfo `num_qt` field
max_num_publishers	uint32	4	PriceInfo `num` field
attestation_time	int64	8	Unix timestamp from this price attestation; Based on Solana attester contract call time
publish_time	int64	8	PriceInfo `publish_time` field
prev_publish_time	int64	8	PriceInfo `prev_publish_time` field
prev_price	int64	8	PriceInfo `prev_price` field
prev_conf	uint64	8	PriceInfo `prev_conf` field

Appendix: Changes since version 2

For readers familiar with the previous version of this format, we list a brief summary of the changes introduced by this specification.

Conventions

- New commitment: We will increase `minor_version` in headers with each backward-compatible format change (as per append-only fields convention known from v2). We will accompany each new `major_version` spec with resetting `minor_version` to 0. Optionally, minor revisions may be documented with a brief explanation of changes over last known minor revision, in lieu of a complete specification.

Header

- `version` is renamed to `major_version` with unchanged meaning - this adds consistency with `minor_version`
- New `minor_version` and `hdr_size` fields
- `hdr_size` marks the count of remaining header bytes. `payload_id` is the only header field that comes after.

PriceAttestation

- `header` is removed
- Fields `price_type` and `corp_act` are removed
- `timestamp` is renamed to `attestation_time` with unchanged meaning and moved past `max_num_publishers`
- `confidence_interval` is renamed to `conf` and moved right after `price`
- `ema_price` and `ema_conf` fields change types from `Rational` to plain 8-byte `int64` and `uint64`, respectively.
- New fields (in order after `status`):
  - `num_publishers` - uint32
  - `max_num_publishers` - uint32
  - *former timestamp → attestation_time - int64*
  - `publish_time` - int64
  - `prev_publish_time` - int64
  - `prev_price` - int64
  - `prev_conf` - uint64