https://sawtooth.hyperledger.org/docs/core/releases/latest/\_autogen/sdk\_submit\_tutorial\_python.html

{

"data": [

{

"batches": [

**Batch 1**

{

1. "header": {

"signer\_public\_key": "0260d6f4c1d83822d1ea01232acfa0161f1b01f9c64a6ef94995ee071ad89cd0fa", -> public key of the client that signed this batch (this will always be the same as the batcher public key in the transaction header)

"transaction\_ids": [

"c7ac4ec8444c5930251442f6e3129c6fbb5b1322ad91d27c54764efb7bd414a378c06adfd543523ed502c88b7e4e4e5580ba6b6a24325c5b842f94b7008d510e" -> list of transaction.header\_signatures that match the order of transactions required for this batch.

]

},

2. "header\_signature": "4282ec7107b2cceb35e4d768ea08562dc333ca335fc61a2d9a338eec6e80c7db3df68398a5fbb0ca8fc43d6ae4d6d56401ab21b8ca0b9064ff2dbaf93641d920", -> Signature that was derived from signing the header

4. "trace": false, -> flag to indicate if this should be traced throughout the system which will result in more debugging output

3. "transactions": [

**Transaction 1 in Batch 1**

{

i. "header": {

"batcher\_public\_key": "0260d6f4c1d83822d1ea01232acfa0161f1b01f9c64a6ef94995ee071ad89cd0fa", -> public key of the client that added this transaction into a batch (this will always be the same as the signer public key in the batch header)

"dependencies": [], -> a list of transaction.header\_signature that describes the transactions that must be processed before this transaction can be processed.

"family\_name": "sawtooth\_settings", -> name of the transaction processor family

"family\_version": "1.0", -> version of the transaction processor family

"inputs": [

"000000a87cb5eafdcca6a8cde0fb0dec1400c5ab274474a6aa82c1c0cbf0fbcaf64c0b", for example for intkey transaction processor, hashlib.sha512('intkey'.encode('utf-8')).hexdigest()[0:6] + hashlib.sha512('first'.encode('utf-8')).hexdigest()[-64:], note that this will vary depending on how the TP is designed.

"000000a87cb5eafdcca6a8cde0fb0dec1400c5ab274474a6aa82c12840f169a04216b7",

"000000a87cb5eafdcca6a8cde0fb0dec1400c5ab274474a6aa82c1918142591ba4e8a7",

"000000a87cb5eafdcca6a8cde0fb0dec1400c5ab274474a6aa82c12840f169a04216b7"

], -> list of addresses that the TP can read from

"nonce": "", -> a random string to provide uniqueness to an otherwise identical transaction

"outputs": [

"000000a87cb5eafdcca6a8cde0fb0dec1400c5ab274474a6aa82c1c0cbf0fbcaf64c0b",

"000000a87cb5eafdcca6a8cde0fb0dec1400c5ab274474a6aa82c12840f169a04216b7"

], -> a list of addresses that the TP can write to

"payload\_sha512": "ee0ae0952107a4f59af7a1809883fa110f02bb0c054985029d7a83d3a424e3cc74e5510fd26c3317c3e9a0c0571442c3fba036a556792a1508222f389f581eb7", a sha512 hash of the encoded payload, for example for intkey transaction processor hashlib.sha512(cbor.dumps({'Verb':'set', 'Name':'first', 'Value':1})).hexdigest(). Note that this will vary depending on how the TP is designed.

"signer\_public\_key": "0260d6f4c1d83822d1ea01232acfa0161f1b01f9c64a6ef94995ee071ad89cd0fa" -> public key of the client that signed the transaction header (note that the one signing the transaction can be different than the one adding the transaction to a batch e.g. offline transactions)

},

ii. "header\_signature": "c7ac4ec8444c5930251442f6e3129c6fbb5b1322ad91d27c54764efb7bd414a378c06adfd543523ed502c88b7e4e4e5580ba6b6a24325c5b842f94b7008d510e", - > signature from signing the header

iii. "payload": "CAESfwomc2F3dG9vdGguc2V0dGluZ3Mudm90ZS5hdXRob3JpemVkX2tleXMSQjAyNjBkNmY0YzFkODM4MjJkMWVhMDEyMzJhY2ZhMDE2MWYxYjAxZjljNjRhNmVmOTQ5OTVlZTA3MWFkODljZDBmYRoRMTUyMjU2OTUxMC44MTM3MDE=" -> encoded family specific information of the transaction, example cbor({‘Verb’: verb, ‘Name’: name, ‘Value’: value}). Note that for any TP, there will be an additional layer of base64 encoding so you will always need to do b64decode before you can decode your encoded payload.

}

**Transaction 2 in Batch 1**

{

.

.

.

}

]

}

**Batch 2**

{

.

.

.

}

],

"header": {

"batch\_ids": [

"4282ec7107b2cceb35e4d768ea08562dc333ca335fc61a2d9a338eec6e80c7db3df68398a5fbb0ca8fc43d6ae4d6d56401ab21b8ca0b9064ff2dbaf93641d920"

],

"block\_num": "0",

"consensus": "R2VuZXNpcw==",

"previous\_block\_id": "0000000000000000",

"signer\_public\_key": "031d1ea8f952e85d858cebde4fc0aaac81c39e36411bed366bdd57c7e942d11c66",

"state\_root\_hash": "de4ef1e3460f65f28856d1eeac63b522da752917370b3afa39225a21e08d654c"

},

"header\_signature": "fca3cfc1ce1aa944e10dc69afd37c3dc111ff8581c485a087611f6ab06c235cd128159880879faf58456a40f23d21482dfd2b5a84f6ed1942f31a57d12bab6f8"

}

],

"head": "fca3cfc1ce1aa944e10dc69afd37c3dc111ff8581c485a087611f6ab06c235cd128159880879faf58456a40f23d21482dfd2b5a84f6ed1942f31a57d12bab6f8", -> id of the head block of the chain the resource was fetched from (particularly useful to know if an explicit head was not set in the original request)

"link": "http://localhost:8008/blocks?head=fca3cfc1ce1aa944e10dc69afd37c3dc111ff8581c485a087611f6ab06c235cd128159880879faf58456a40f23d21482dfd2b5a84f6ed1942f31a57d12bab6f8&start=0x0000000000000000&limit=100", -> a link to the resource fetched with both head and paging parameters explicitly set

"paging": {

"limit": null,

"start": null

} -> information on how the resource was paginated and how further pages can be fetched (https://sawtooth.hyperledger.org/docs/core/releases/1.0.1/architecture/rest\_api.html#data-envelope)