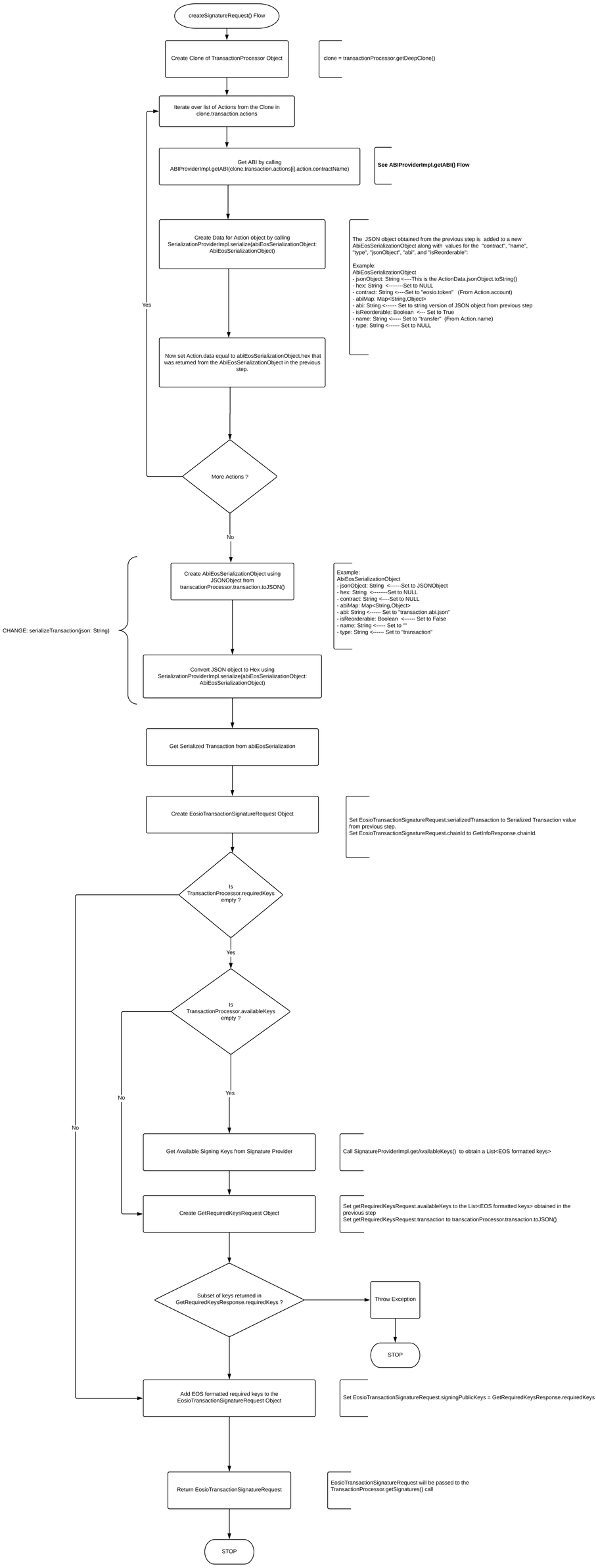
createSignatureRequest() Flow
Create Clone of TransactionProcessor Object
clone = transactionProcessor.getDeepClone()
Iterate over list of Actions from the Clone in clone.transaction.actions
Get ABI by calling ABIProviderImpl.getABI(clone.transaction.actions[i].action.contractName)
See ABIProviderImpl.getABI() Flow
Create Data for Action object by calling SerializationProviderImpl.serialize(abiEosSerializationObject: AbiEosSerializationObject)
The JSON object obtained from the previous step is added to a new AbiEosSerializationObject along with values for the "contract", "name", "type", "jsonObject", "abi", and "isReoderable":
Example:
AbiEosSerializationObject
- jsonObject: String <----This is the ActionData.jsonObject.toString()
- hex: String <--------Set to NULL
- contract: String <----Set to "eosio.token" (From Action.account)
- abiMap: Map<String,Object>
- abi: String <------ Set to string version of JSON object from previous step
- isReorderable: Boolean <--- Set to True
- name: String <----- Set to "transfer" (From Action.name)
- type: String <------ Set to NULL
Yes
Now set Action.data equal to abiEosSerializationObject.hex that was returned from the AbiEosSerializationObject in the previous step.
More Actions ?
No
CHANGE: serializeTransaction(json: String)
Create AbiEosSerializationObject using JSONObject from transcationProcessor.transaction.toJSON()
Example:
AbiEosSerializationObject
- jsonObject: String <------Set to JSONObject
- hex: String <--------Set to NULL
- contract: String <----Set to NULL
- abiMap: Map<String,Object>
- abi: String <------ Set to "transaction.abi.json"
- isReorderable: Boolean <------ Set to False
- name: String <----- Set to ""
- type: String <------ Set to "transaction"
Convert JSON object to Hex using SerializationProviderImpl.serialize(abiEosSerializationObject: AbiEosSerializationObject)
Get Serialized Transaction from abiEosSerialization
Create EosioTransactionSignatureRequest Object
Set EosioTransactionSignatureRequest.serializedTransaction to Serialized Transaction value from previous step.
Set EosioTransactionSignatureRequest.chainId to GetInfoResponse.chainId.
Is TransactionProcessor.requiredKeys empty ?
Yes
Is TransactionProcessor.availableKeys empty ?
Yes
No
No
Get Available Signing Keys from Signature Provider
Call SignatureProviderImpl.getAvailableKeys() to obtain a List<EOS formatted keys>
Create GetRequiredKeysRequest Object
Set getRequiredKeysRequest.availableKeys to the List<EOS formatted keys> obtained in the previous step
Set getRequiredKeysRequest.transaction to transcationProcessor.transaction.toJSON()
Subset of keys returned in GetRequiredKeysResponse.requiredKeys ?
Throw Exception
STOP
Add EOS formatted required keys to the EosioTransactionSignatureRequest Object
Set EosioTransactionSignatureRequest.signingPublicKeys = GetRequiredKeysResponse.requiredKeys
Return EosioTransactionSignatureRequest
EosioTransactionSignatureRequest will be passed to the TransactionProcessor.getSignatures() call
STOP