
DeCipher
Algorand ABI

Application Binary Interface

DeCipher 2021

Speakers:

Jason Weathersby
Sr Director Developer Relations
Algorand
Twitter: @JasonWeathersby

DeCipher 2021

Algorand ABI

The ABI Specification

Using with Goal

Using with the SDK

New Opcodes

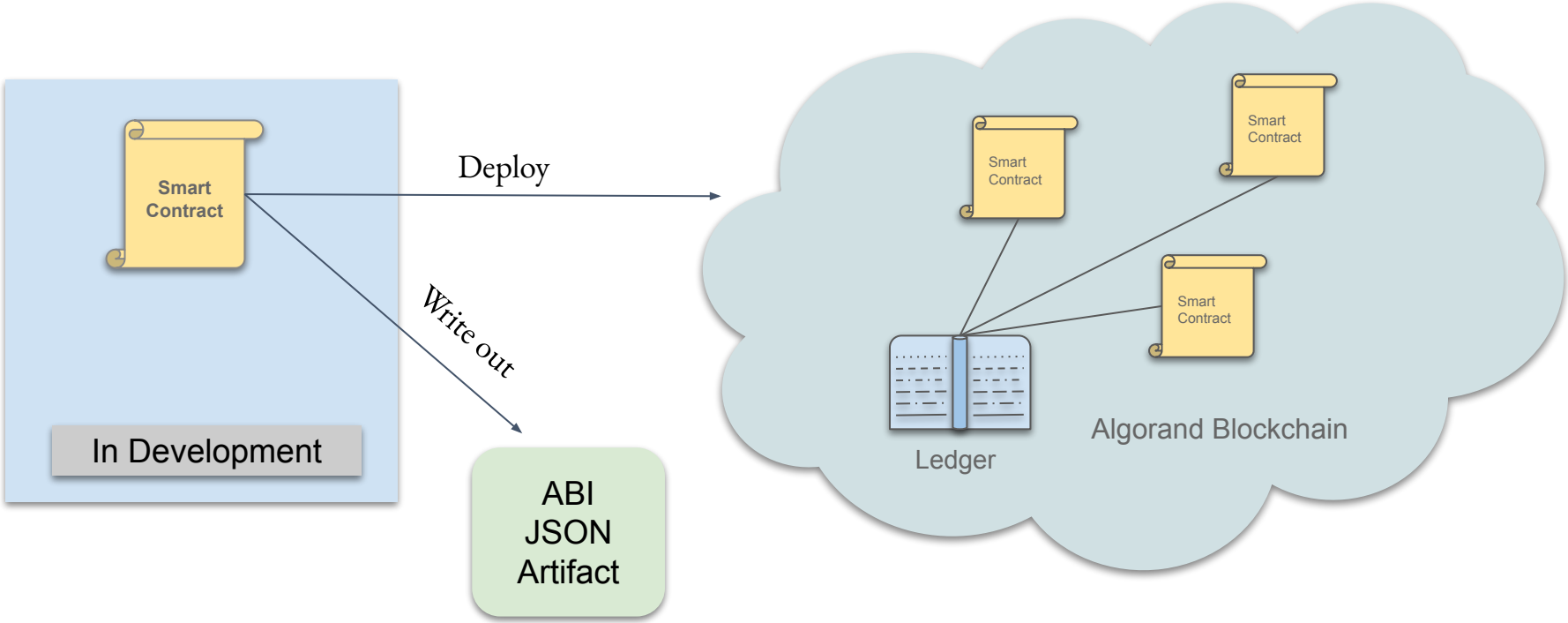
ABI Specification

ARC-04

- Specification for interoperability between smart contracts and applications
- Defines standard for encoding contract calls to smart contracts.
- Includes method signatures that contain arguments and return types
- Allows wallets and other dapps to construct app calls based on a description of the interface
- SDKS support consuming a JSON file with the description

ABI Overview

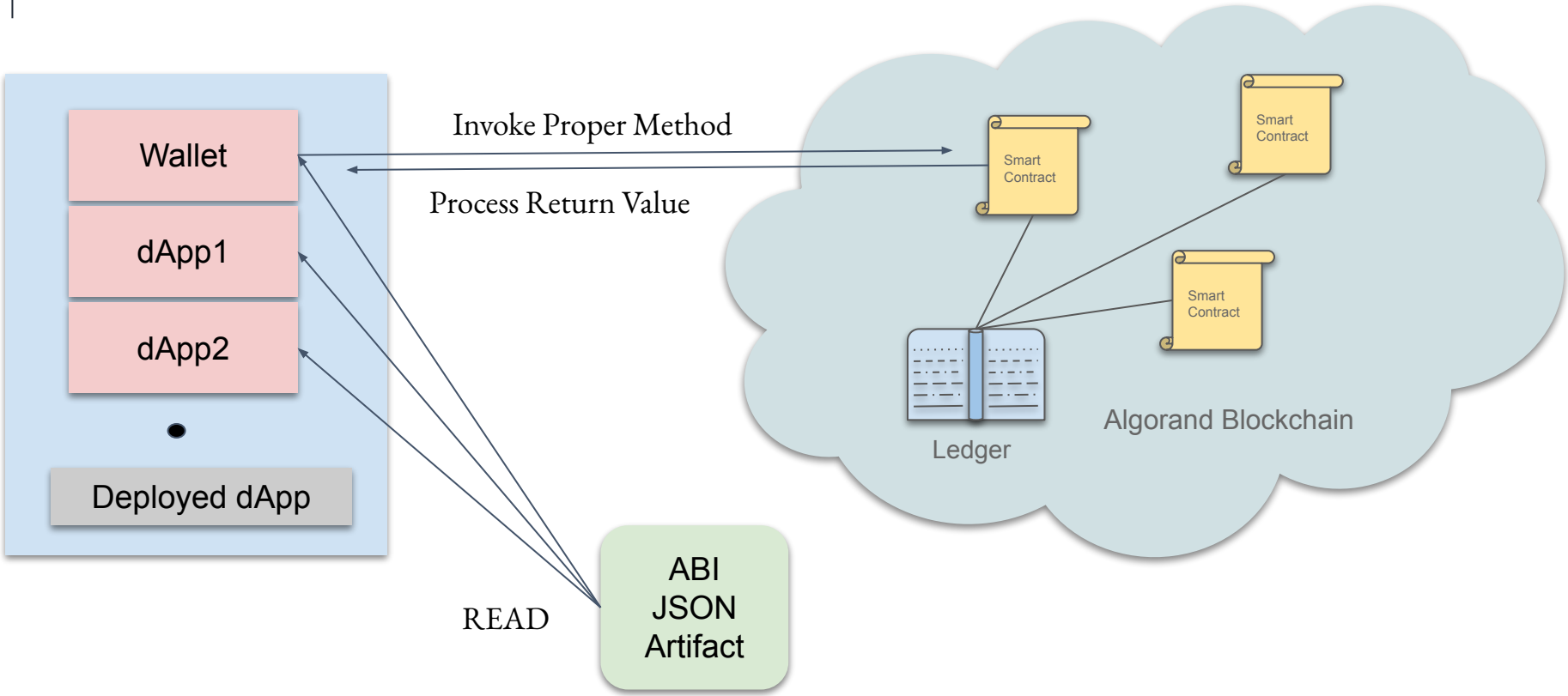
DeCipher 2021



Without the JSON, dApps have no way to know how to call your contract or understand return values, without reading the code

ABI Overview

DeCipher 2021



ABI Overview - Specification

DeCipher 2021

- Defines the JSON Artifact formatting
- Defines the Method invocation rules
- Specifies the argument and return types allowed
- Defines encoding rules Goal and the SDKs use encode and decode method names, arguments and return values
- Defines Three Top Level JSON artifacts
    - Method
    - Interface
    - Class
- Not Mandatory, but **highly** recommended

ABI
JSON
Artifact

ABI Overview - Method JSON

DeCipher 2021

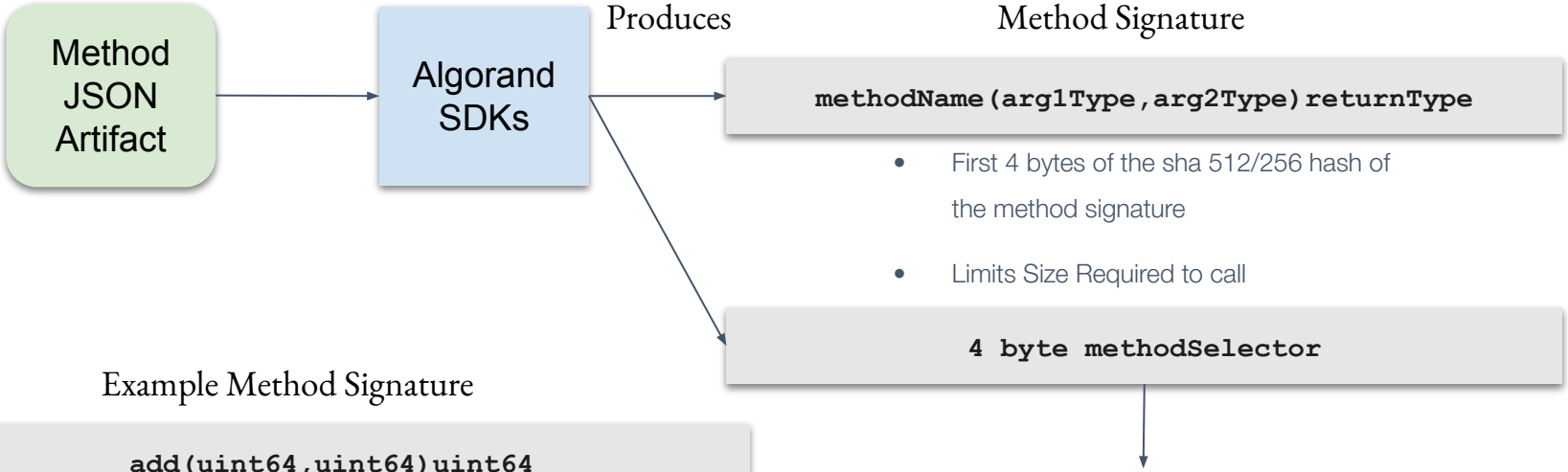
```
interface Method {
  name: string,
  desc?: string,
  args: Array<{ name?: string, type: string, desc?: string }>,
  returns: { type: string, desc?: string }
}
```

```
{
   "name":"add",
   "desc":"Add 2 integers",
   "args":[ { "type":"uint64","name":"parm1", "desc":"first parameter" }, { "type":"uint64" } ],
   "returns": {"type":"uint64"}
}
```

- Method names should start with a letter followed by any number of characters that are either letters,numbers or underscores.
- Camel case is preferred
- Optional data can be used by wallets and dApps for display and understanding purposes

ABI Overview - Method Signatures

DeCipher 2021



Example Method Signature

```
add(uint64,uint64)uint64
```

Example Method Selector

```
hash (hex): 8aa3b61f0f1965c3a1cbfa91d46b24e54c67270184ff89dc114e877b1753254a

Selector:

8aa3b61f
```

ABI Overview - Interface JSON

DeCipher 2021

```
interface Interface {
  name: string,
  methods: Array<Method>
}
```

```
{
   "name":"my interface",
   "methods":[
       {
           "name":"add",
           "args":[{"type":"uint32"},{"type":"uint32"}],
           "returns":{"type":"uint32"}
       }]
}
```

- Logical collection of methods
- Each method must generate a unique method selector
- A smart contract implements an interface if it supports all methods

ABI Overview - Contract JSON

DeCipher 2021

```
interface Contract {
  name: string,
  appId: number
  methods: Array<Method>
}
```

```
{
   "name":"contract",
   "appId":123,
   "methods":  [{"name":"add",
               "args":[{"type":"uint32"},{"type":"uint32"}],
               "returns":{"type":"uint32"}}
               ]
}
```

- Similar to an interface, but represents a concrete deployed contract
- Each method must generate a unique method selector
- Complete set of methods a contract implements
- May be said that it implements an Interface with possible additional methods
- Contract may contain special `_optIn` and `_closeOut` methods
- If not used and local state is used in a call, developer may set for method
    - onComplete: algosdk.OnApplicationComplete.OptInOC
    - onComplete: algosdk.OnApplicationComplete.CloseOutOC,
    - If not set and local state is used, call will fail

ABI Types - Method Signatures

DeCipher 2021

Types
uint<N> - N-> 8-512
byte
bool
address
string
ufixed<N>x<M>
<type>[]
<type>[<N>]
(T1,T2,....TN) - Tuple

N bit unsigned fixed decimal point with M precision

Foreign Types
account
asset
application

Only in arguments, not in tuples or arrays

Special Types
txn -any txn
pay
keyreg
acfg
axfer
afrz
aapl

Used to specify additional Txes that method needs

Dynamic Types
[]

Will most likely require extract*

opcodes to process in Teal

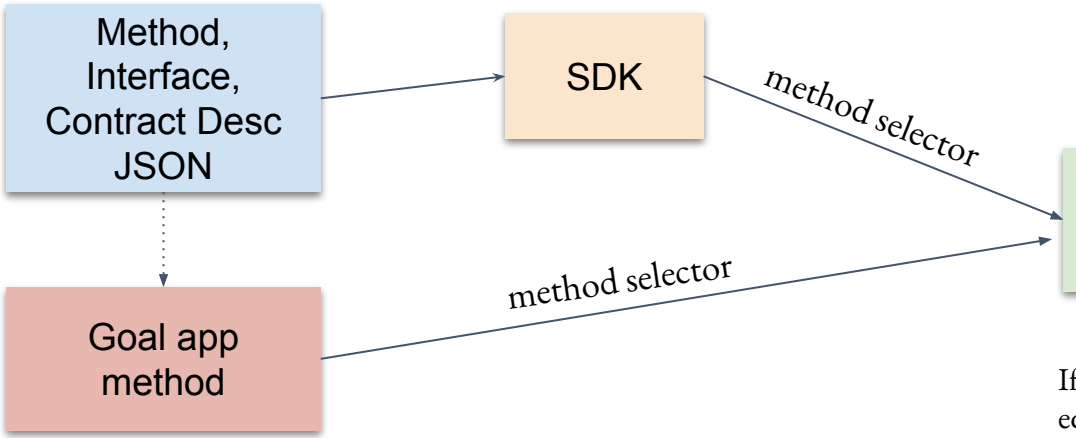
Method Invocation

Standard Format

- First app transaction parameter should be method selector
  - Accessed in Teal as `txna ApplicationArgs 0`
  - SDKS and Goal handle
- Next forteen parameters are method parameters
- The fifteen parameter can contain a tuple of additional parameters
  - SDKs do this automatically
  - PyTeal will handle in future
  - TEAL programs need to use extract for the tuple
- Return value is returned using unique log with 4 byte prefix
  - prefix hash of the word return
  - Last log in the transaction with this prefix

App Developer Description

DeCipher 2021



If SDK Method Selector is not equal to the TEAL selector the method will never be called

- Switch on selector using method opcode
- Return values with log opcode and 4 byte prefix

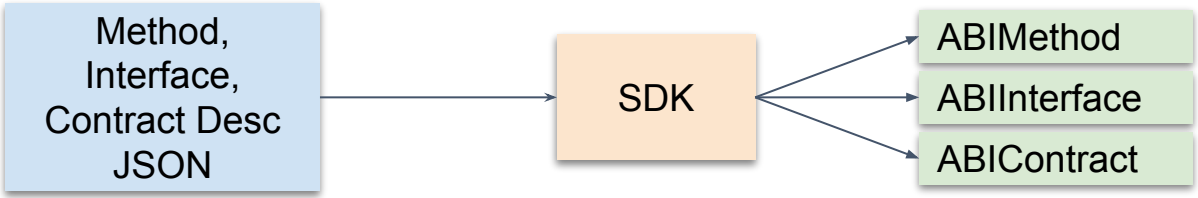
DeCipher 2021

Demo

Goal method call

SDK JSON Descriptions

DeCipher 2021



DeCipher 2021

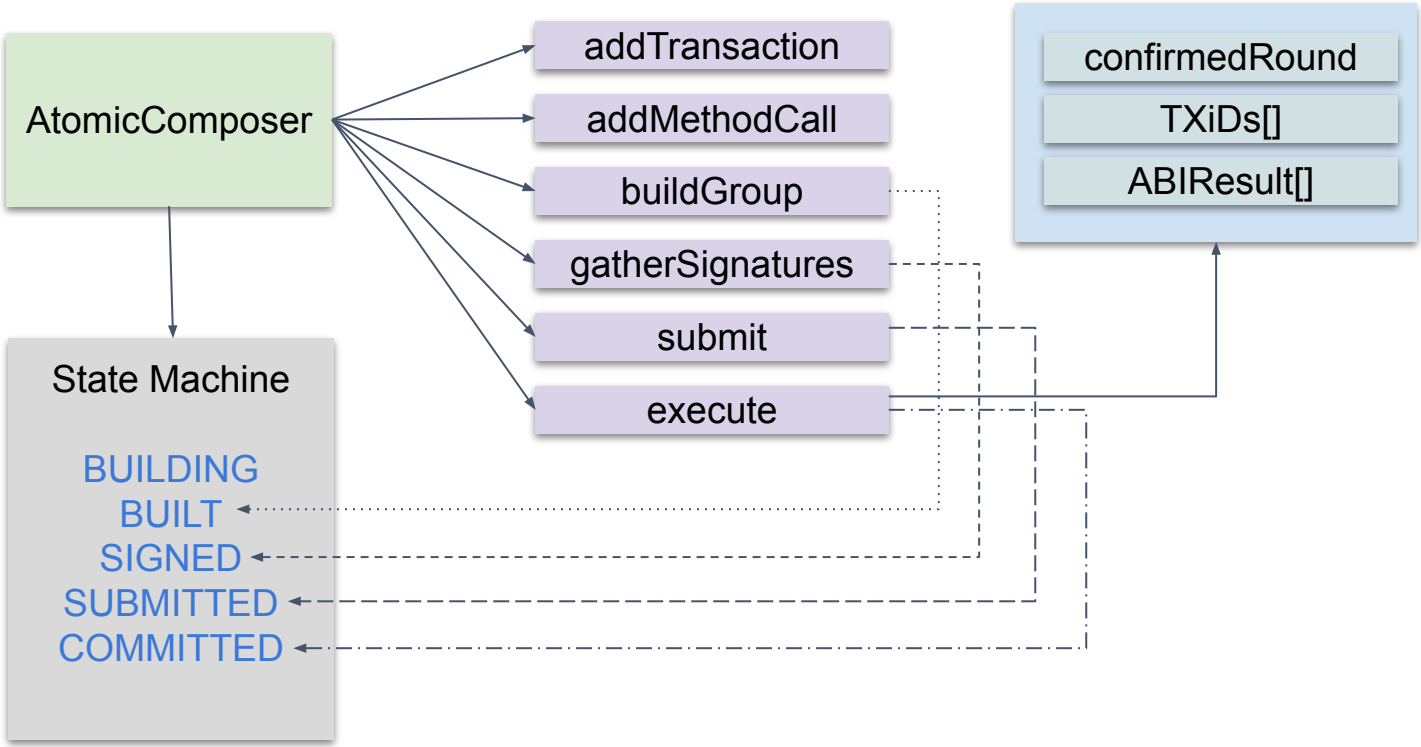
Demo

JSON to JS SDK

SDK New Atomic Builder

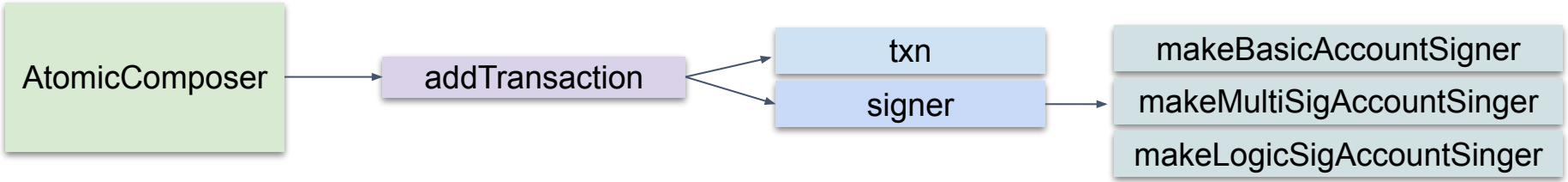
DeCipher 2021

- One or More Transactions
- For any transaction
- Enforces ABI



SDK New Atomic Builder addTransaction

DeCipher 2021




```
let txn = algosdk.makePaymentTxnWithSuggestedParams(....
let transactionWithSigner = {
    txn: txn,
    signer: algosdk.makeBasicAccountTransactionSigner(acct),
};
comp.addTransaction(transactionWithSigner)
```

SDK New Atomic Builder addMethodCall

DeCipher 2021


```
const sp = await client.getTransactionParams().do()
const commonParams = {
    appId:contract.appId,
    sender:acct.addr,
    suggestedParams:sp,
    signer: algosdk.makeBasicAccountTransactionSigner(acct)
}
const comp = new algosdk.AtomicTransactionComposer()
comp.addMethodCall({
    method: sum, methodArgs: [1,1], ...commonParams
})
```

DeCipher 2021

Demo

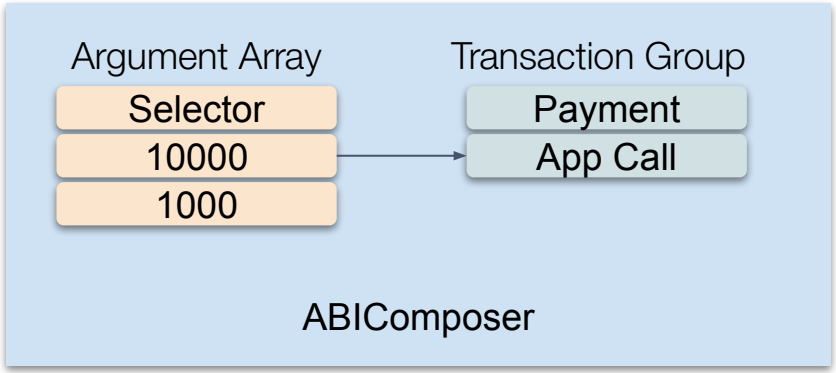
JS SDK ABIComposer: Simple Payment and Method Call

Method Descriptions and Special Types

DeCipher 2021

```
myCall(pay,uint64,uint64)uint64
```

```
let txn = algosdk.makePaymentTxnWithSuggestedParams(acct.addr, acct.addr, 10000, undefined, undefined, sp);
comp.addMethodCall({
    method: myCall,
    methodArgs: [
        {
            txn: txn,
            signer: algosdk.makeBasicAccountTransactionSigner(acct)
        },
        10000,
        1000
    ],
    ...commonParams
})
```

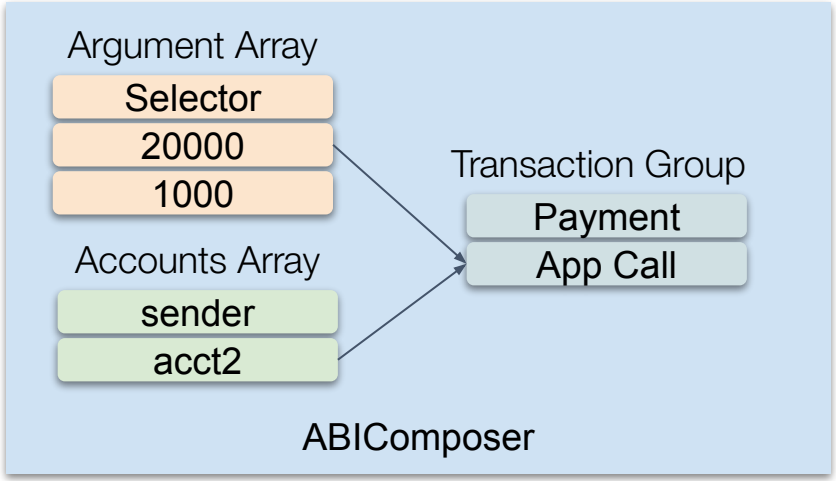


Method Descriptions and Foreign Types

DeCipher 2021

```
myCall(pay,uint64,uint64,account)uint64
```

```
let txn = algosdk.makePaymentTxnWithSuggestedParams(acct.addr, acct.addr, 10000, undefined, undefined,
sp);
comp.addMethodCall({
    method: myCall,
    methodArgs: [
        {
            txn: txn,
            signer: algosdk.makeBasicAccountTransactionSigner(acct)
        },
        20000,
        1000,
        acct2.addr
    ],
    ...commonParams
})
```



DeCipher 2021

Demo

JS SDK
ABIComposer: multi transaction calls

DeCipher 2021

Smart Contract Requirements

method, log and extract opcodes

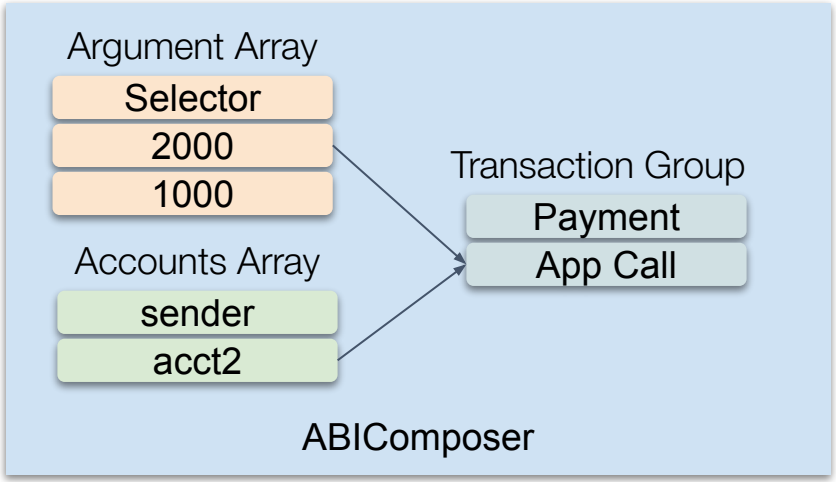
Method Opcode

DeCipher 2021

```
myCall(pay,uint64,uint64,account)uint64
```

- method opcode converts signature to selector
- Use method opcode to compare
- Make sure to cover selectors that match no method

```
method "myCall(pay,uint64,uint64,account)uint64"
txna ApplicationArgs 0
==
bnz myCall_Sub
```



Example Return values

DeCipher 2021

Example PyTeal Return value for an integer

```
Log(
    Concat(
        Bytes("base16", "0x151f7c75"), #Literally hash('return')[:4]
        Itob(b)
    )
)
```

Example Teal Return value for an integer

```
store 0 //store result
byte 0x151f7c75 //return bytes
load 0 //containing return integer
itob
concat
log
```

Example PyTeal Return value for a byte array

```
Log(
    Concat(
        Bytes("base16", "0x151f7c75"), #Literally hash('return')[:4]
        b
    )
)
```

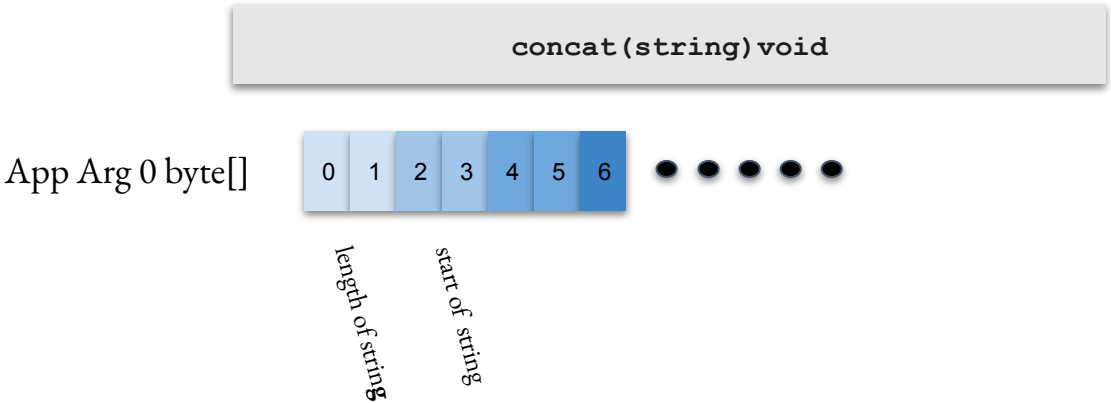
Example Teal Return value for an byte[]

```
store 0 //store result
byte 0x151f7c75 //return bytes
load 0 //containing return byte[]
concat
log
```

- only required on non-void return values
- last log with the 4-byte signature
- PyTeal will change to simplify

Dynamic Types Extract Opcode

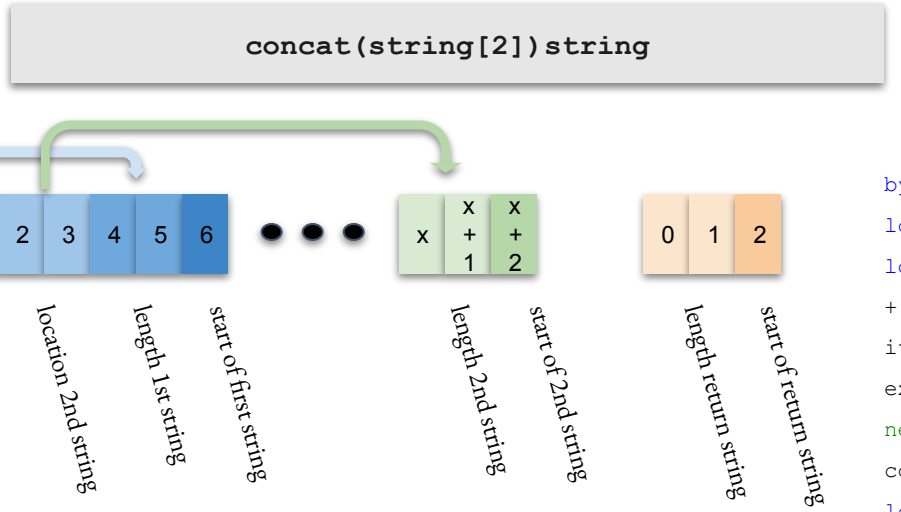
DeCipher 2021




```
txna ApplicationArgs 1
int 0
extract_uint16 //length of first string
```

Dynamic Types Extract Opcode

DeCipher 2021



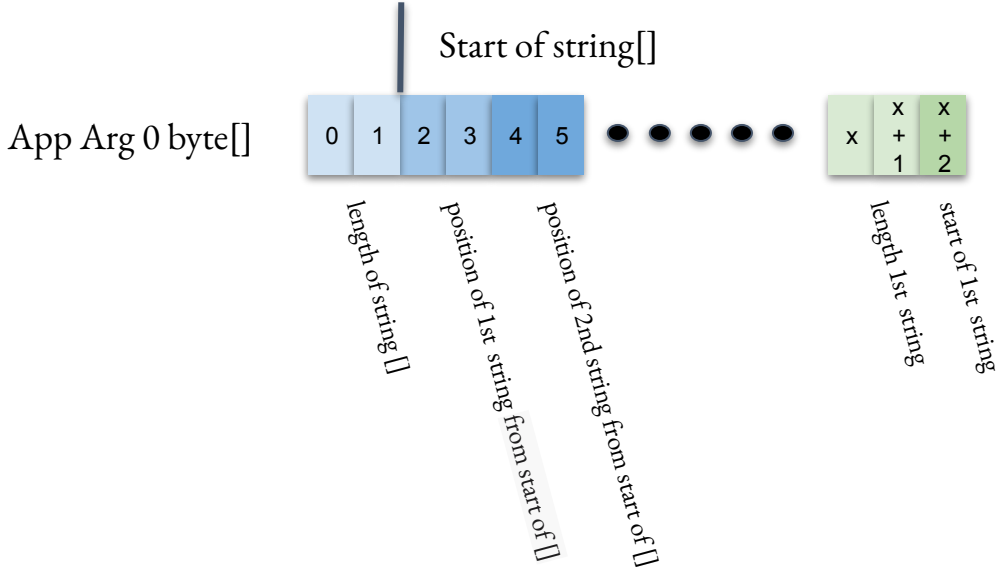

```
txna ApplicationArgs 1
int 0
extract_uint16 //location of first string
```

```
byte 0x151f7c75 //return bytes
load 11 //length of first string
load 13 //length of second string
+
itob
extract 6 2 //itob produces 8 bytes we just need the last two
concat
load 0 //containing return contated strings
concat
log
```

Extract Opcode

DeCipher 2021

```
concat(string[])void ie [][]
```




```
txna ApplicationArgs 1
int 0 //index into argument byte[]
extract_uint16 //length of array
```

DeCipher 2021

Demo

TEAL: method opcode and returning values

DeCipher 2021

More Examples

https://github.com/algorand-devrel/demo-abi

DeCipher 2021

Questions?