Algorand

Algorand AVM 1.0 Smart Contracts

Agenda

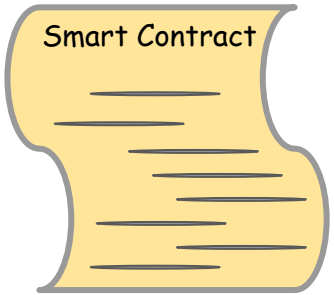
- Overview of Smart Contracts and Smart Signatures
- Transaction Execution Approval Language (TEAL)
- Developing Smart Contracts with Python

Algorand Smart Contracts and Smart Signatures

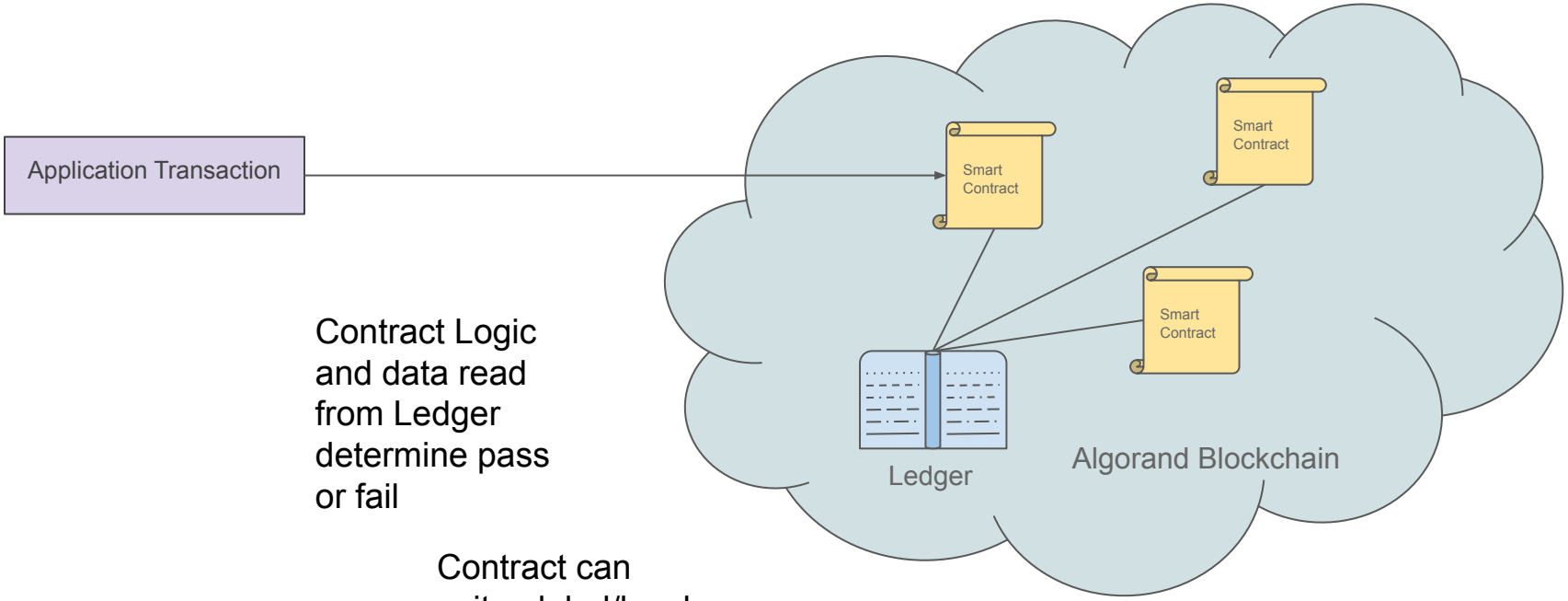
- **Smart Contracts and Smart Signatures**
  - Smart Contracts - On-chain Global and Local Storage
    - Escrow Accounts
    - On-chain dApps
  - Smart Signatures - Used to sign transactions
    - Delegate

- Combinable with Other Algorand Technology
  - Atomic Transfers
  - Algorand Assets
  - Combine Smart Contracts with Smart Signatures

- **Transaction Execution Approval Language**
  - The contract logic is written in TEAL
  - Python Enabled Compiler (PyTEAL)



Smart Contract High Level

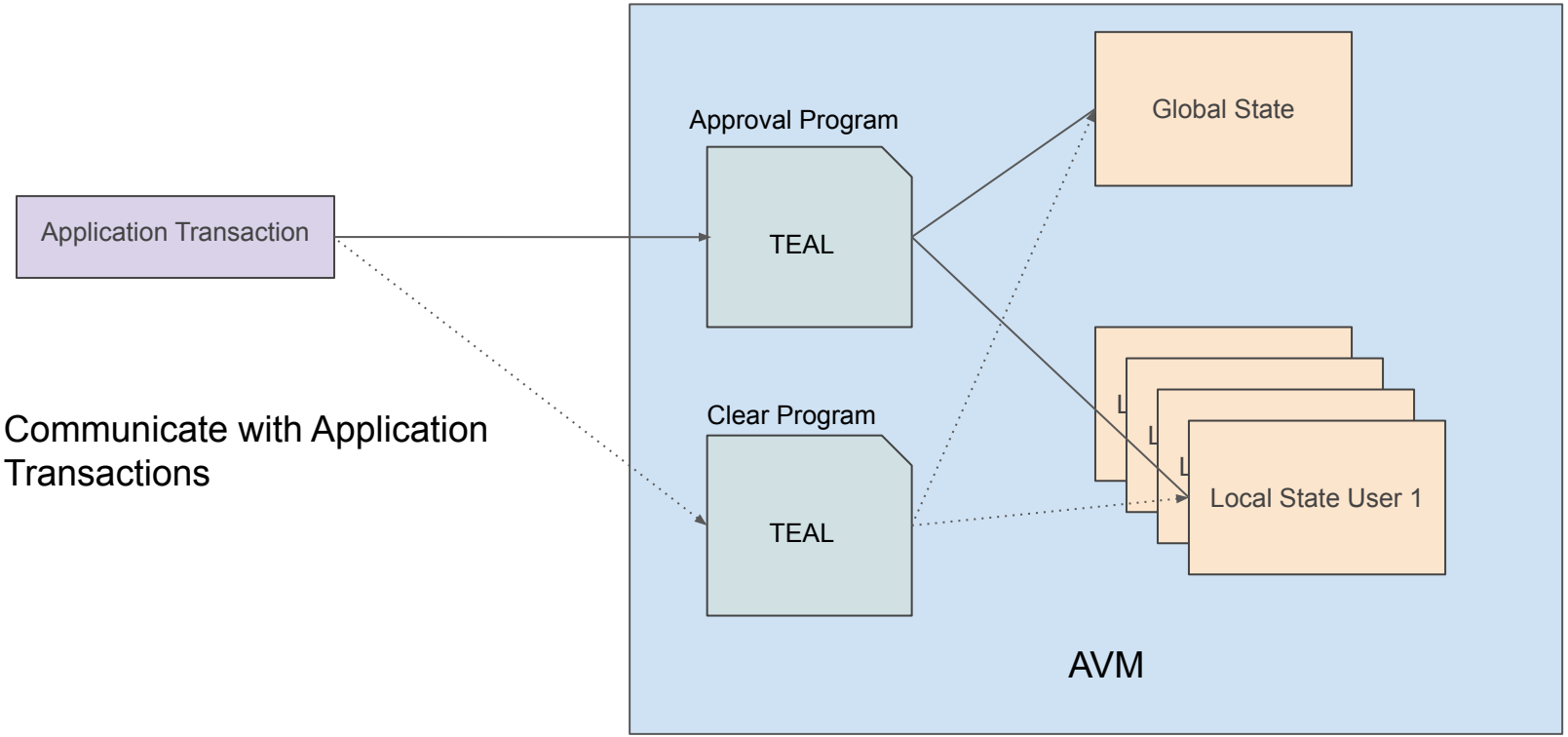


Contract Logic and data read from Ledger determine pass or fail

Contract can write global/local values to the ledger

Contract can also initiate new transactions

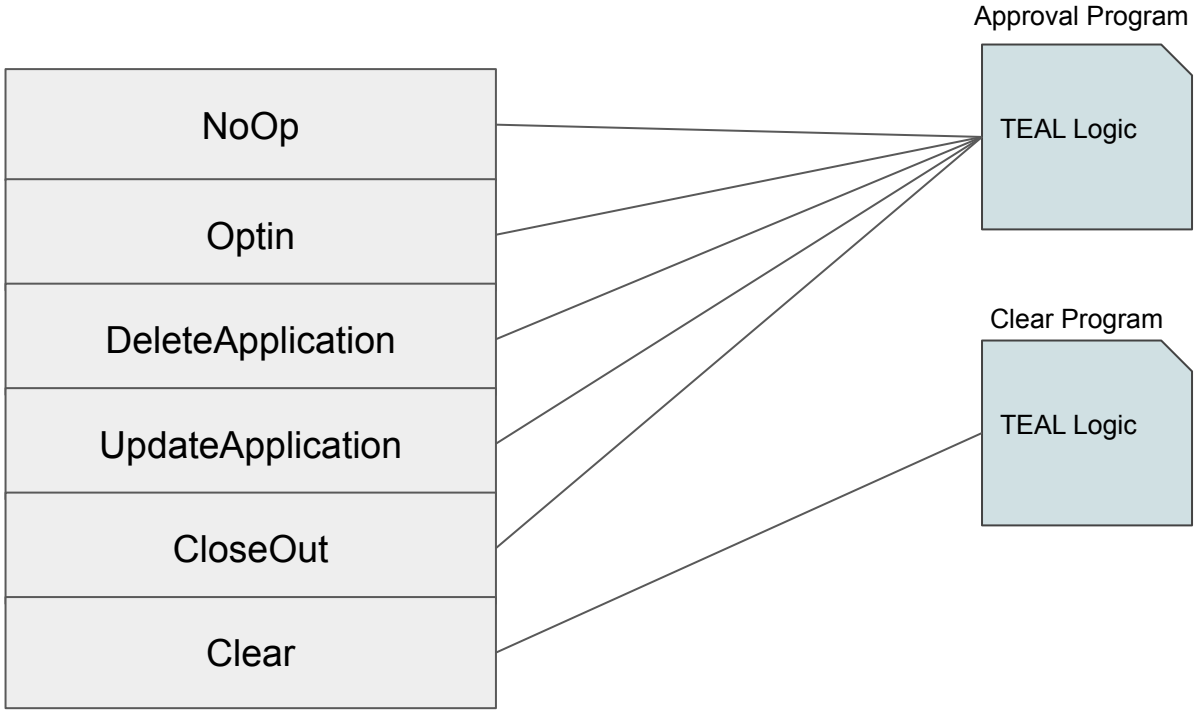
Smart Contracts aka Apps



Communicate with Application Transactions

Transaction Sub-Types for Application

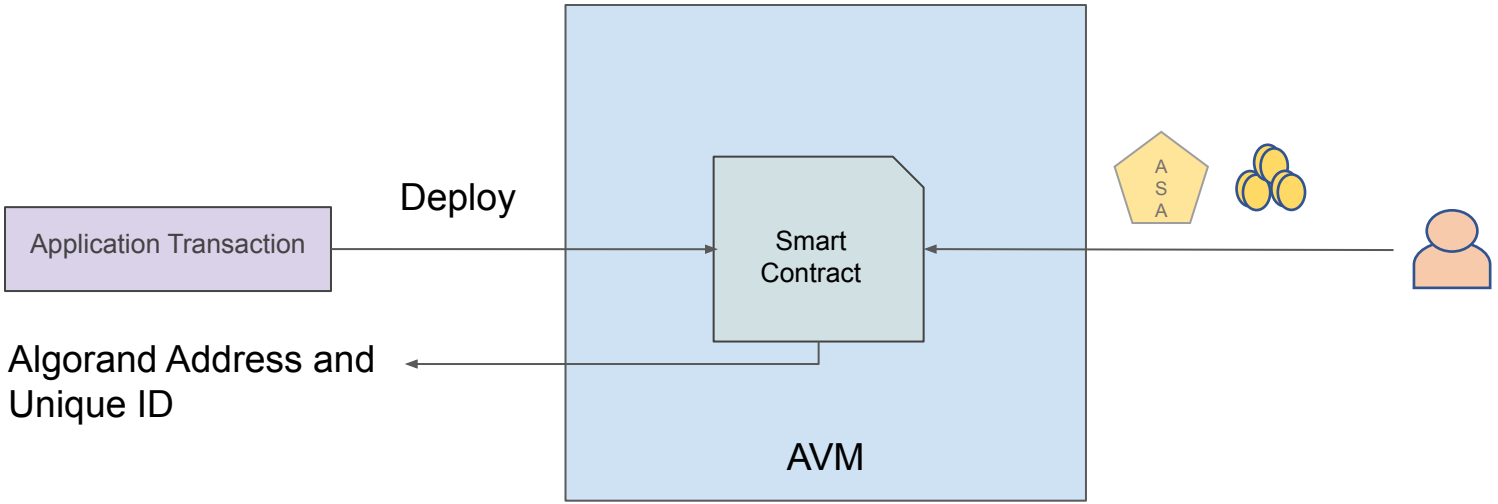
Used to Communicate With Smart Contract



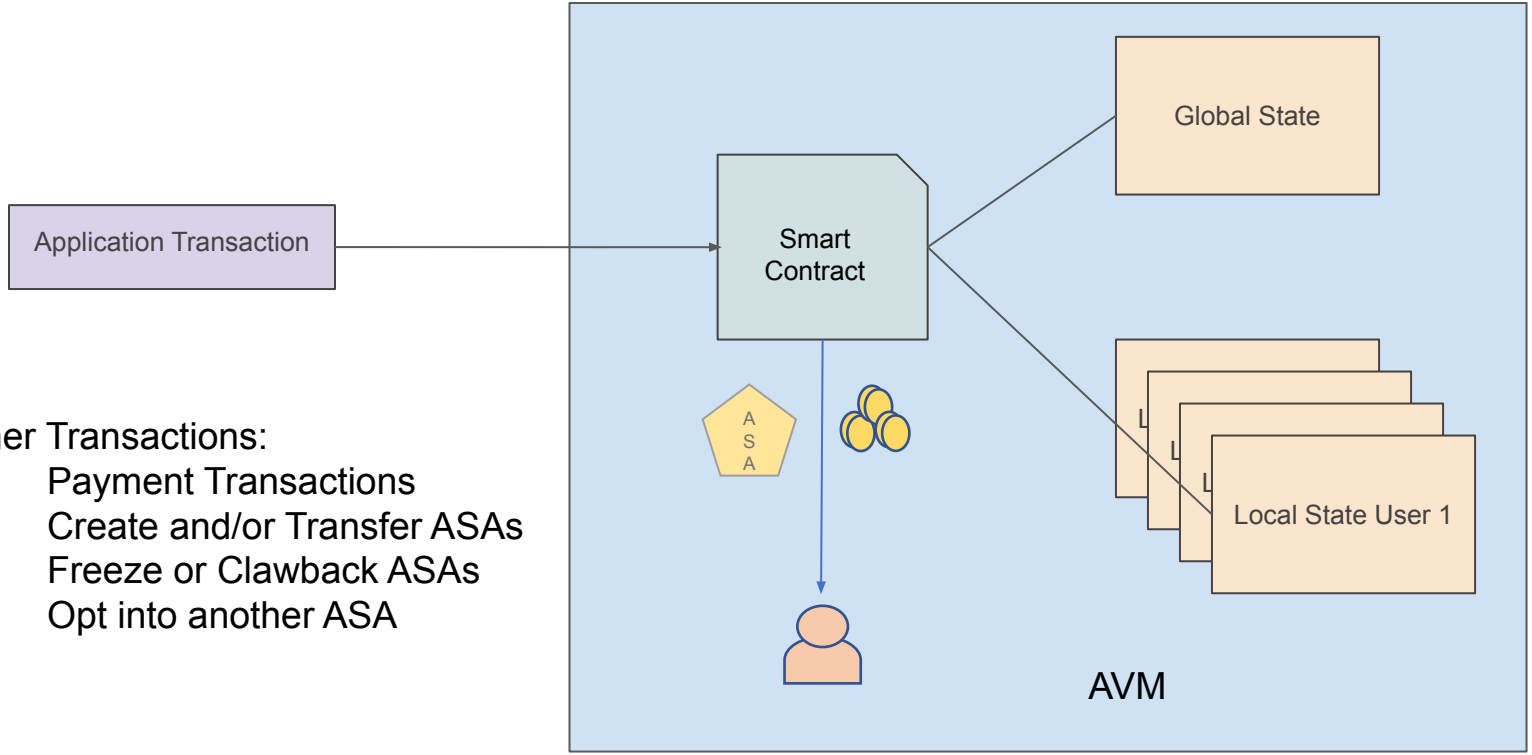
Graceful

Non-Graceful
ie Will clear
regardless

Smart Contract Escrow



Smart Contract Escrow



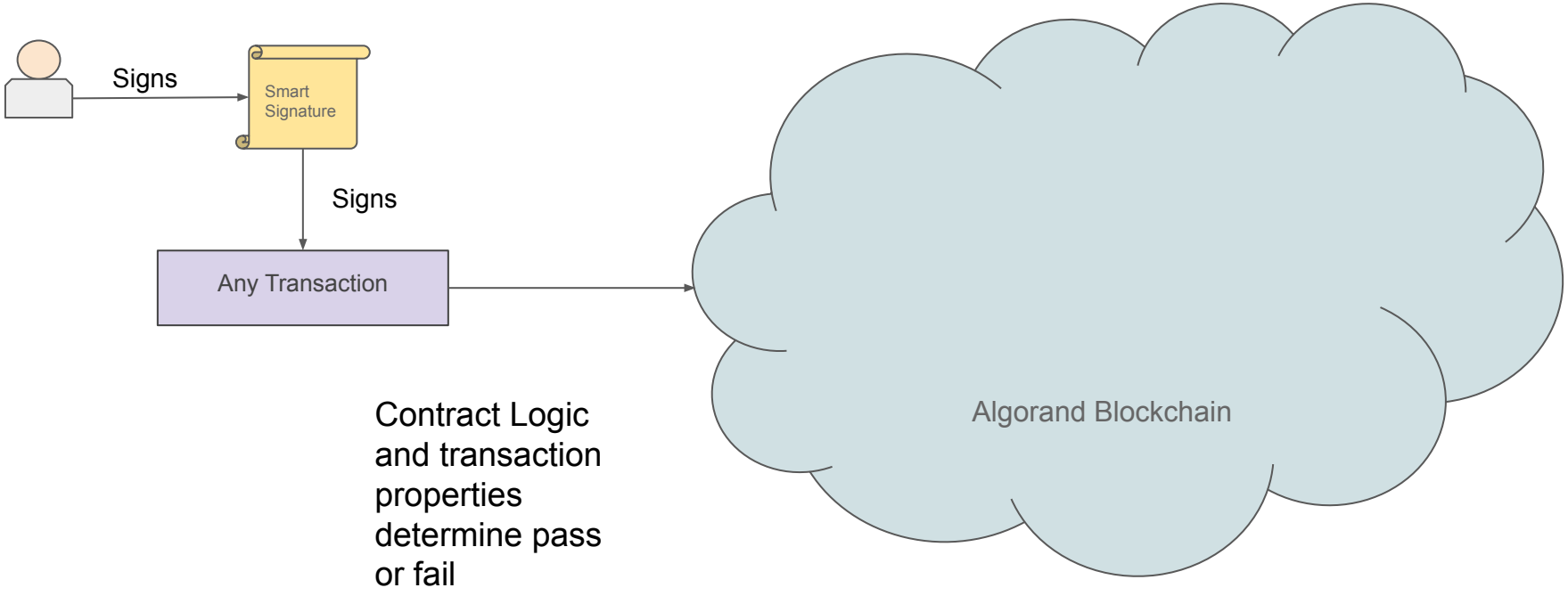
Inner Transactions:

- Payment Transactions
- Create and/or Transfer ASAs
- Freeze or Clawback ASAs
- Opt into another ASA

Stateful Contract Usages

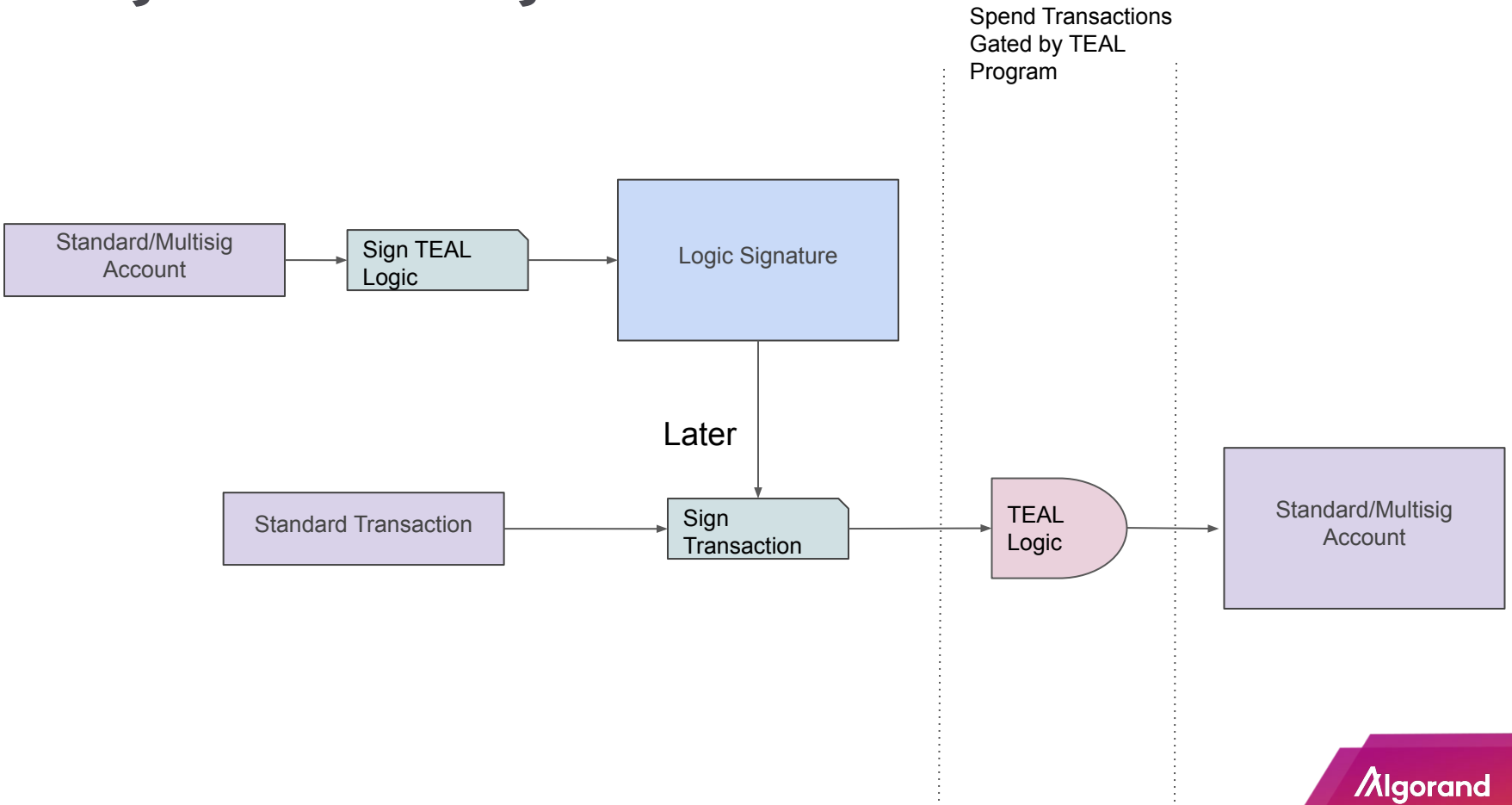
- Escrows
- Defi Apps (AMM, lending, etc)
- Crowdfunding
- Voting
- Material Tracking
- Lottery
- Seating
- Anywhere a global variable is needed
- Anywhere a individual users values need to be manipulated

Smart Signature High Level

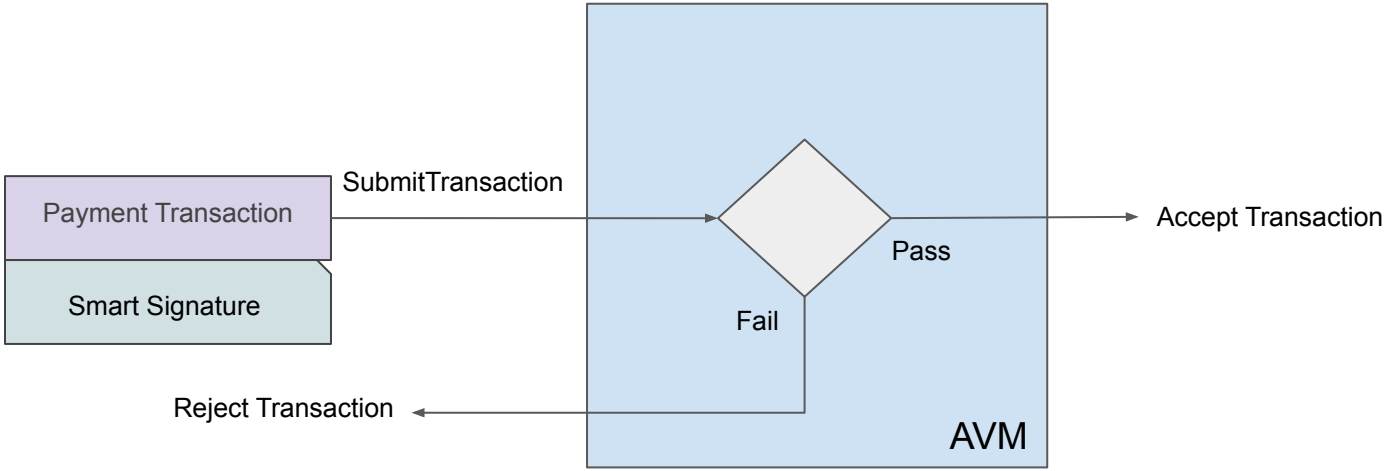


Contract Logic and transaction properties determine pass or fail

Delegate Smart Signature



Smart Signature



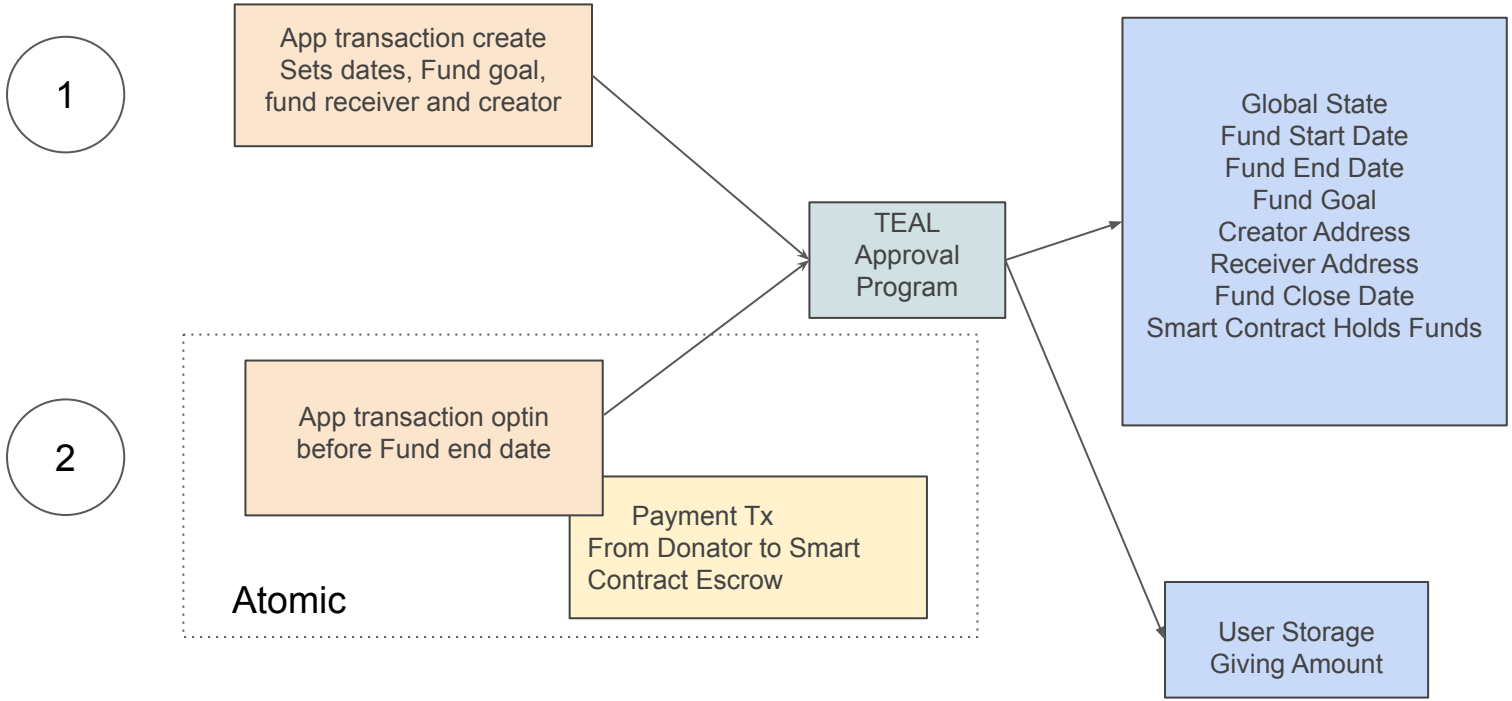
Smart Signature Usages

- **Delegated**
  - Recurring Payments
  - Limited Account Authority

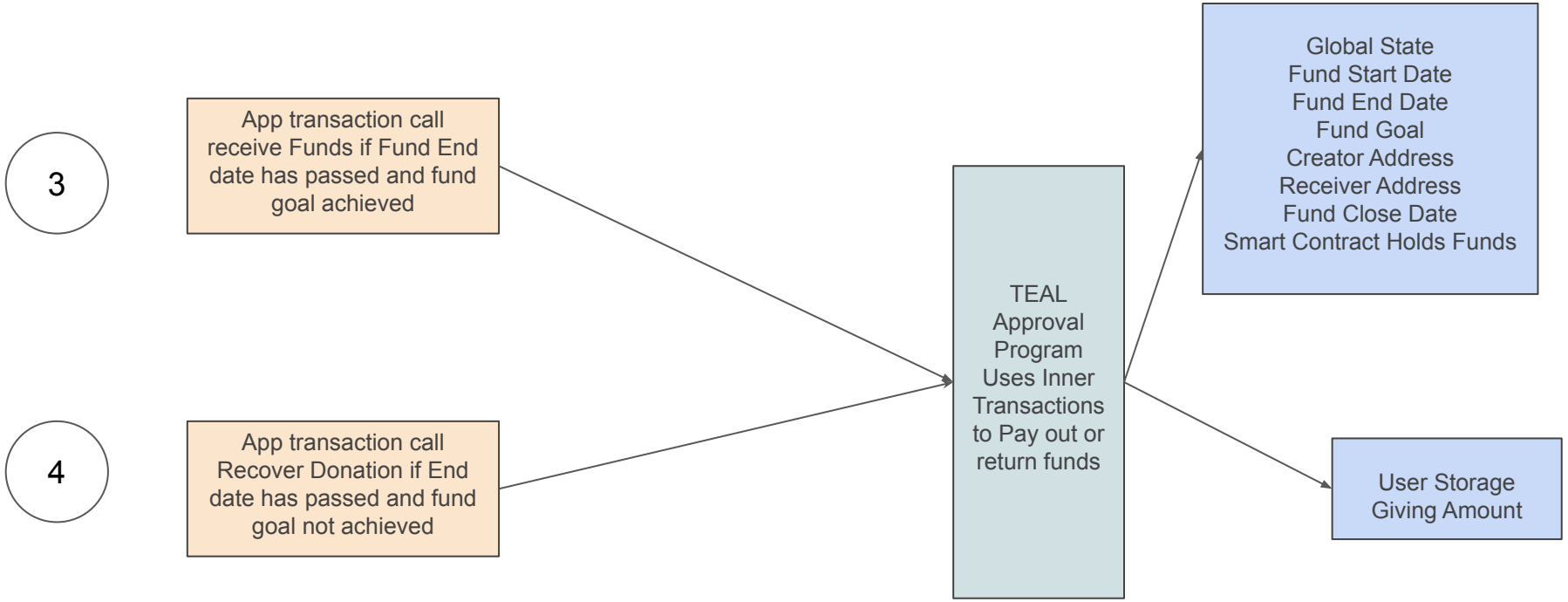
Algorand

Combining Algorand Technologies

Combining Technologies - Crowdfunding



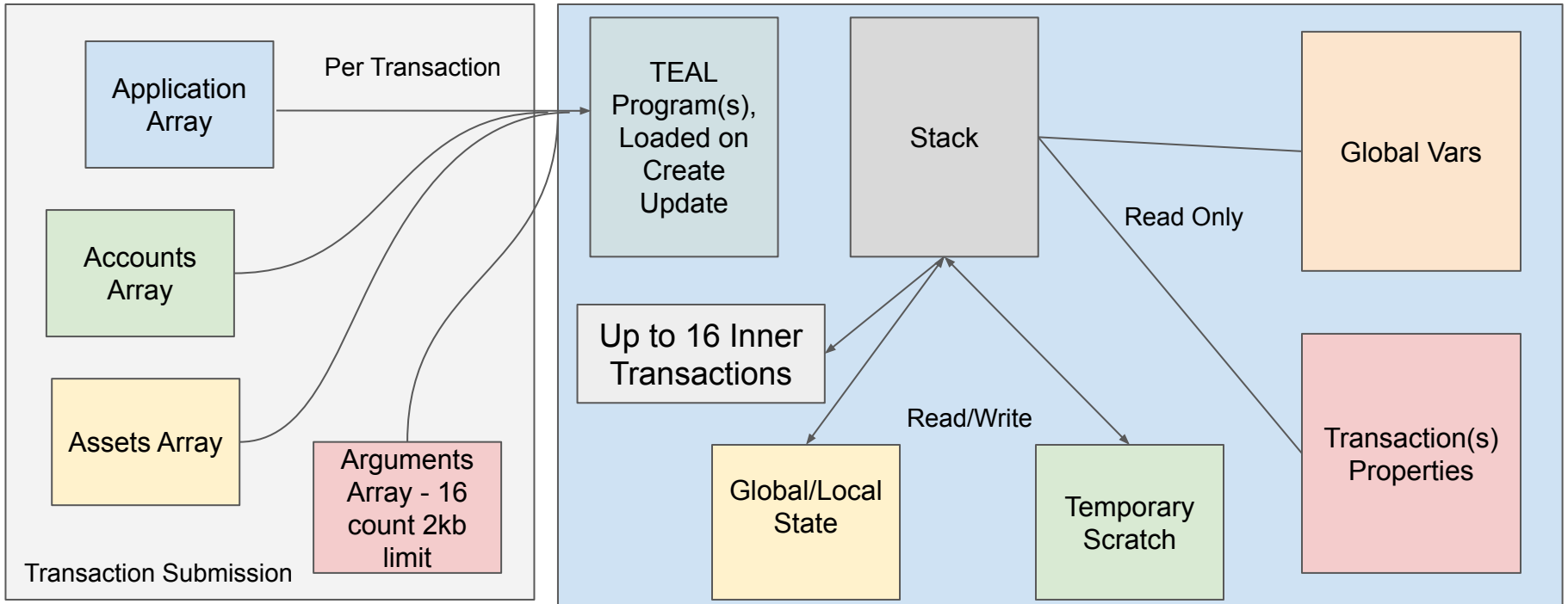
Payout


3
App transaction call receive Funds if Fund End date has passed and fund goal achieved
4
App transaction call Recover Donation if End date has passed and fund goal not achieved
TEAL Approval Program Uses Inner Transactions to Pay out or return funds
Global State
Fund Start Date
Fund End Date
Fund Goal
Creator Address
Receiver Address
Fund Close Date
Smart Contract Holds Funds
User Storage
Giving Amount

Algorand

Runtime Architecture

Smart Contract Runtime Architecture



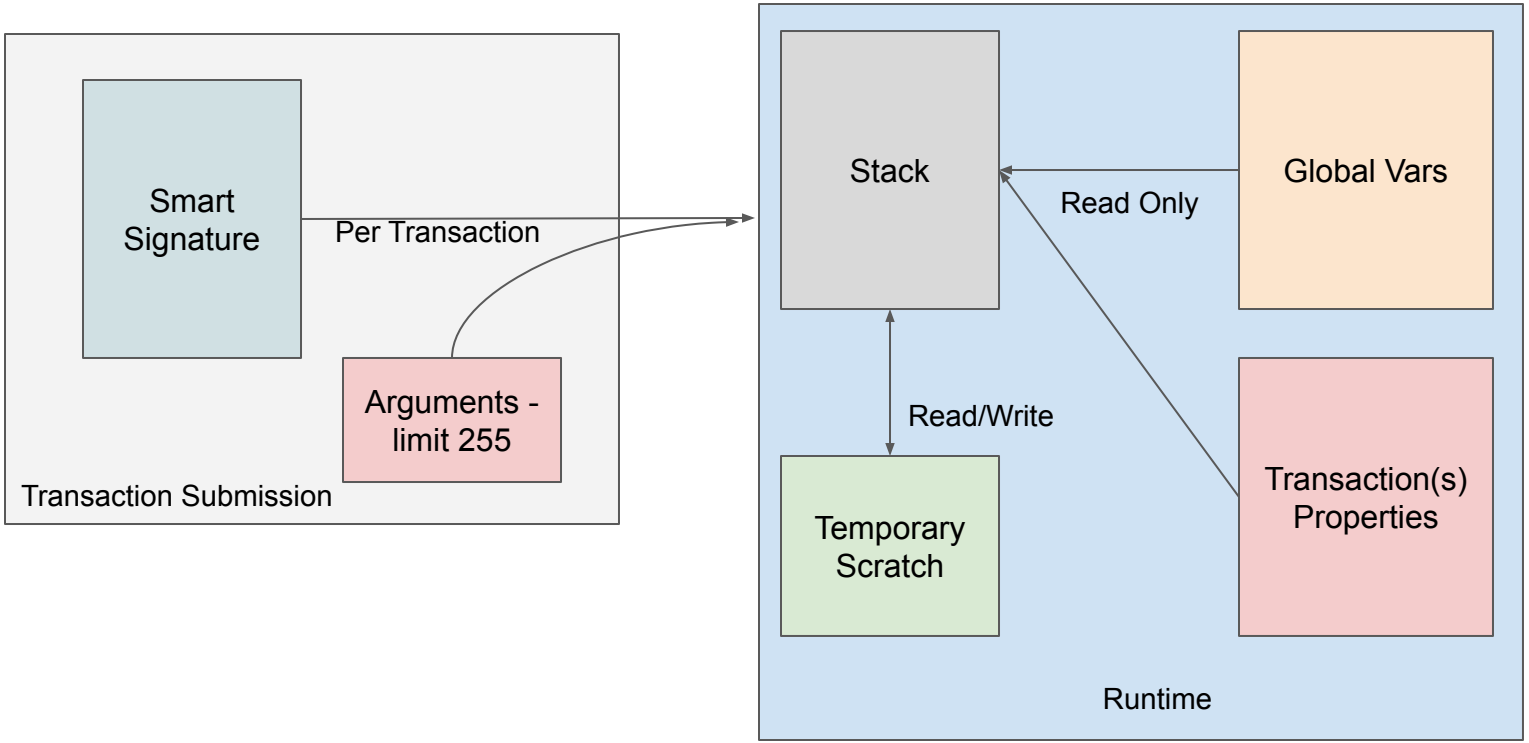
Up to 4 accounts and 8 total for Accounts, Application, Asset Arrays

Global State - 64 KV pairs - 128 total bytes per KV, key limited to 64 bytes

Local State - 16 KV pairs - 128 total bytes per KV, key limited to 64 bytes

Algorand

Smart Signature Runtime Architecture

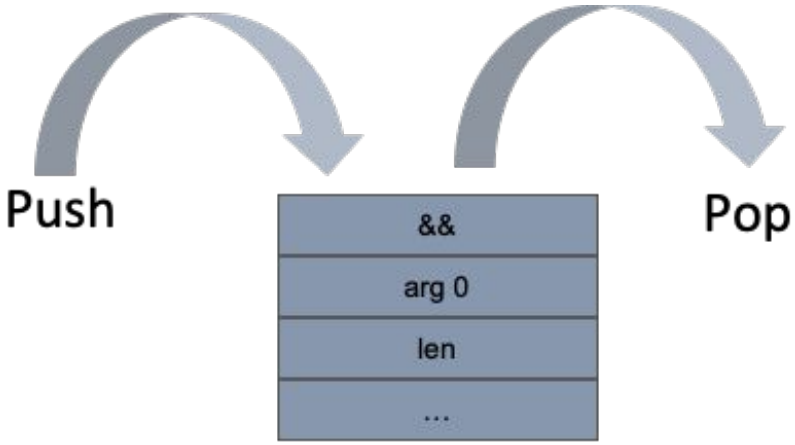


Algorand

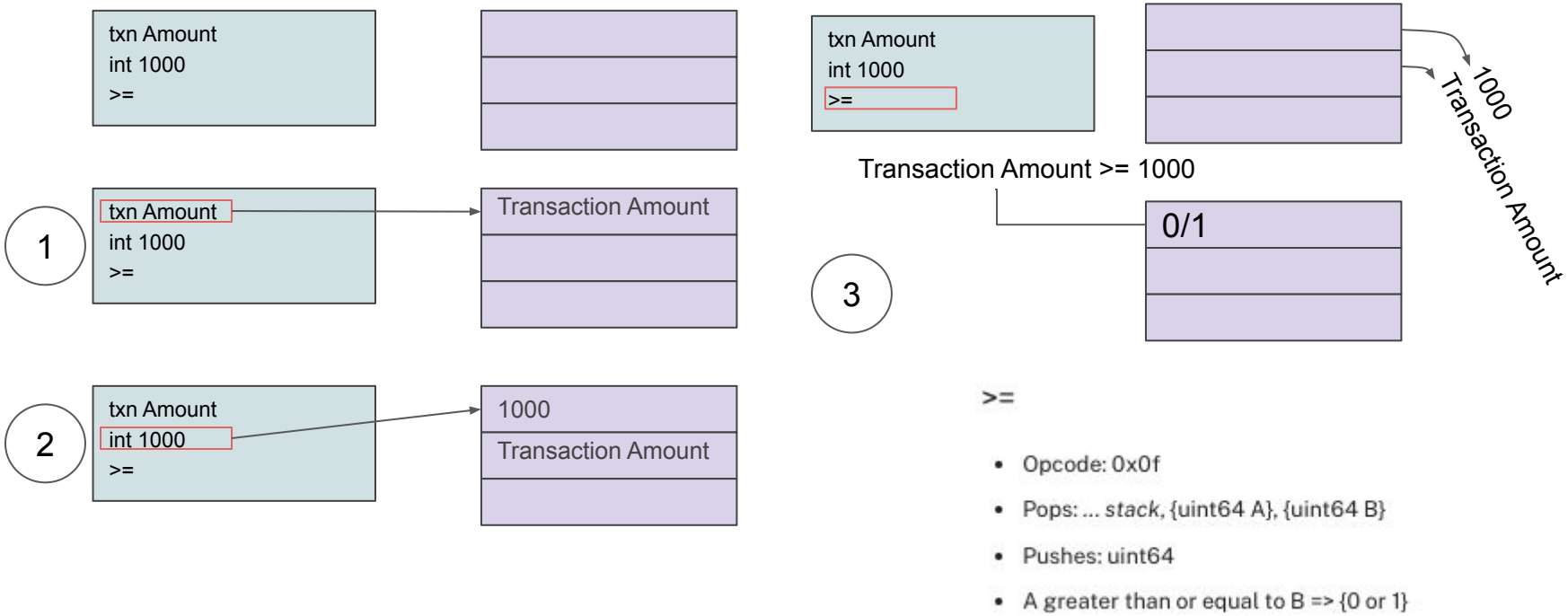
Transaction Execution Appoval Language (TEAL)

TEAL - Transaction Execution Approval Language

- Bytecode based stack language
- Turing Complete
- Looping and Subroutines
- Returns True or False
- SDK Support
- > 130 Opcodes
- Access to ASA/Algo Balances
- Read all Transactions in a Group
- Smart Contract - Global/Local
- Smart Contract - 16 inner transactions
  - Asset transfers, creation, deletion
  - Asset configuration, revoking and freezing
  - Payment transfers
- PyTEAL library to write in python



Simple Stack Example



>=

- Opcode: 0x0f
- Pops: ... *stack*, {uint64 A}, {uint64 B}
- Pushes: uint64
- A greater than or equal to B => {0 or 1}

Opcodes

app_local_get

- Opcode: 0x62
- Pops: ... *stack*, {uint64 A}, {[]byte B}
- Pushes: any
- read from account specified by Txn.Accounts[A] from local state of the current application key B => value
- LogicSigVersion >= 2
- Mode: Application

params: account index, state key. Return: value. The value is zero if the key does not exist.

Opcode Reference Document

Pseudo Operators

int //load an int onto stack

byte //Load bytes on stack

addr //Load Algorand address

TEAL Approval Program

```
txn Receiver
addr HJLWACXDBOEH25KJB2WI2X5BHQOJ4LS2MRLNMHJ5ZZOBLJU7KGDJSHEU3I
==
.
byte “mystring”
txn ApplicationArgs 0
==
```

Algorand

State Demo

Accessing Transaction Properties

- Sender
- Fee
- FirstValid
- FirstValidTime
- LastValid
- Note
- Lease
- Receiver
- Amount
- CloseRemainderTo
- VotePK
- SelectionPK
- VoteFirst
- VoteLast
- VoteKeyDilution
- Type
- TypeEnum
- XferAsset
- AssetAmount
- AssetSender
- AssetReceiver
- AssetCloseTo
- GroupIndex
- TxID
- ...

**over 60 properties

TEAL Approval Program

```
txn Amount
int 1000
>=
```

Global Variables

Index	Name	Type	Notes
0	MinTxnFee	uint64	micro Algos
1	MinBalance	uint64	micro Algos
2	MaxTxnLife	uint64	rounds
3	ZeroAddress	[]byte	32 byte address of all zero bytes
4	GroupSize	uint64	Number of transactions in this atomic transaction group. At least 1
5	LogicSigVersion	uint64	Maximum supported TEAL version. LogicSigVersion >= 2.
6	Round	uint64	Current round number. LogicSigVersion >= 2.
7	LatestTimestamp	uint64	Last confirmed block UNIX timestamp. Fails if negative. LogicSigVersion >= 2.
8	CurrentApplicationID	uint64	ID of current application executing. Fails in LogicSigs. LogicSigVersion >= 2.
9	CreatorAddress	[]byte	Address of the creator of the current application. Fails if no such application is executing. LogicSigVersion >= 3.
10	CurrentApplicationAddress	[]byte	Address that the current application controls. Fails in LogicSigs. LogicSigVersion >= 5.
11	GroupID	[]byte	ID of the transaction group. 32 zero bytes if the transaction is not part of a group. LogicSigVersion >= 5.

Checking Type of Transaction

Value	Constant name	Description
0	unknown	Unknown type. Invalid transaction
1	pay	Payment
2	keyreg	KeyRegistration
3	acfg	AssetConfig
4	axfer	AssetTransfer
5	afrz	AssetFreeze
6	appl	ApplicationCall

TEAL Approval Program

```
txn TypeEnum
int appl
==
```

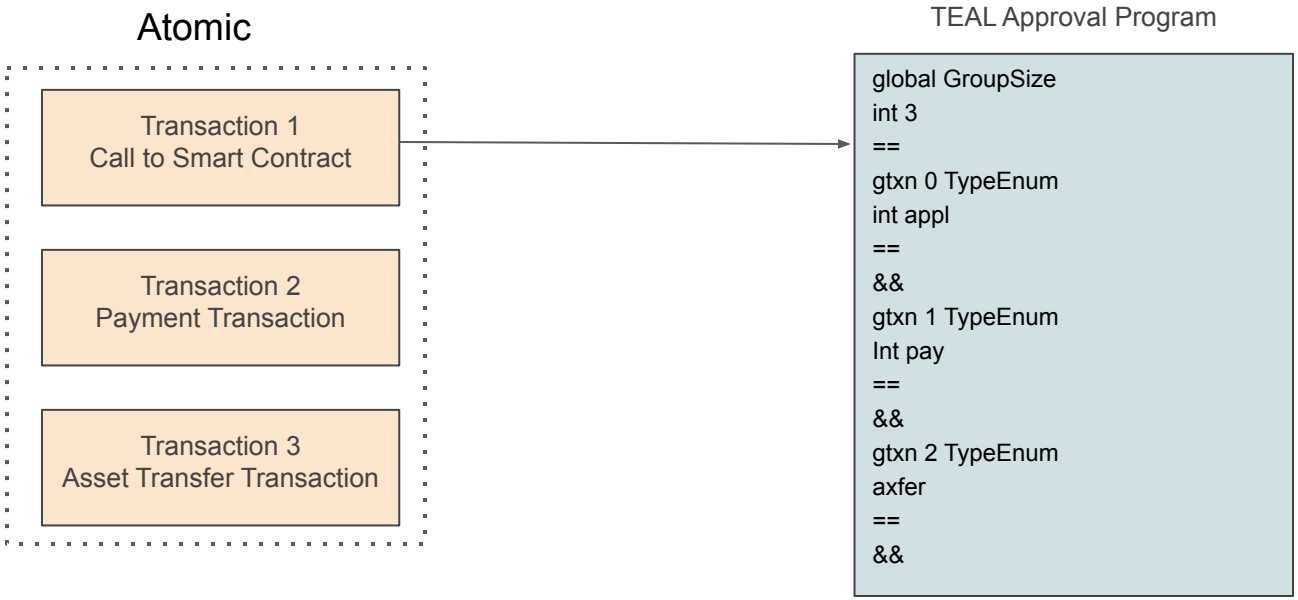
Application Transaction Sub-Types - Stateful

0	NoOp	Only execute the `ApprovalProgram` associated with this application ID, with no additional effects.
1	OptIn	Before executing the `ApprovalProgram`, allocate local state for this application into the sender's account data.
2	CloseOut	After executing the `ApprovalProgram`, clear any local state for this application out of the sender's account data.
3	ClearState	Don't execute the `ApprovalProgram`, and instead execute the `ClearStateProgram` (which may not reject this transaction). Additionally, clear any local state for this application out of the sender's account data as in `CloseOutOC`.
4	UpdateApplication	After executing the `ApprovalProgram`, replace the `ApprovalProgram` and `ClearStateProgram` associated with this application ID with the programs specified in this transaction.
5	DeleteApplication	After executing the `ApprovalProgram`, delete the application parameters from the account data of the application's creator.

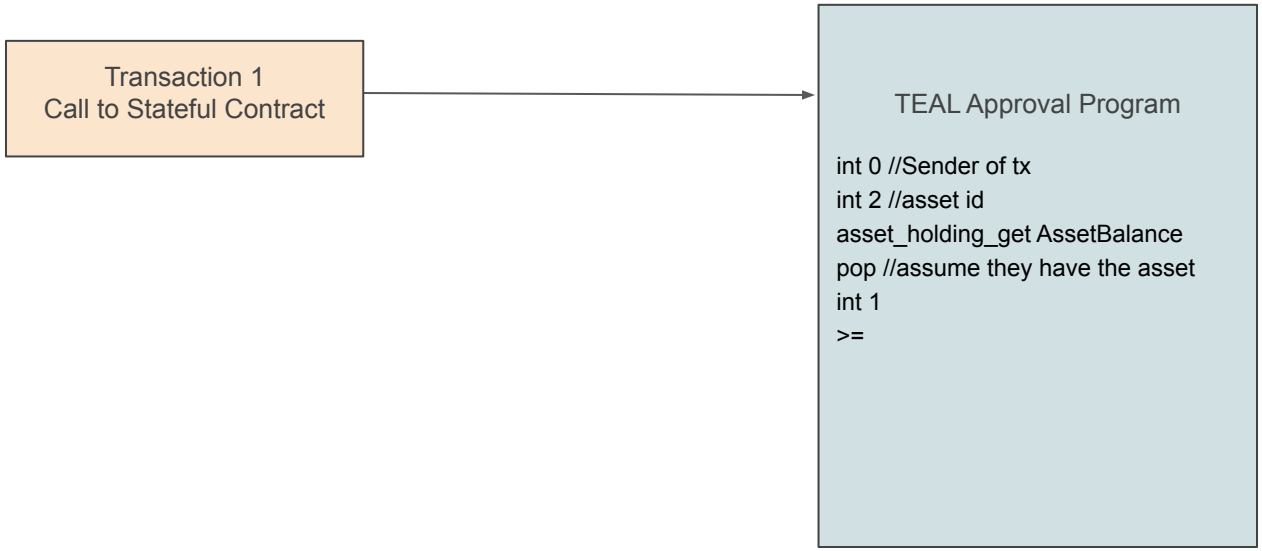
TEAL Approval Program

```
txn OnCompletion
int NoOp
==
```

Atomic Transactions - gtxn vs txn

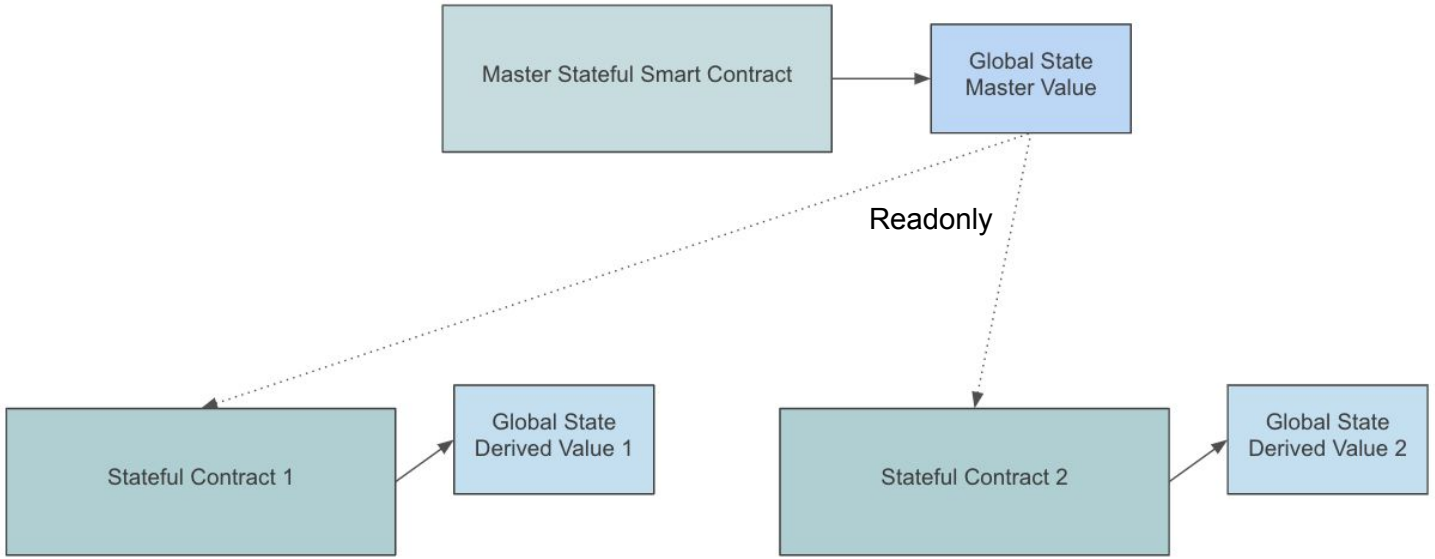


Asset Check



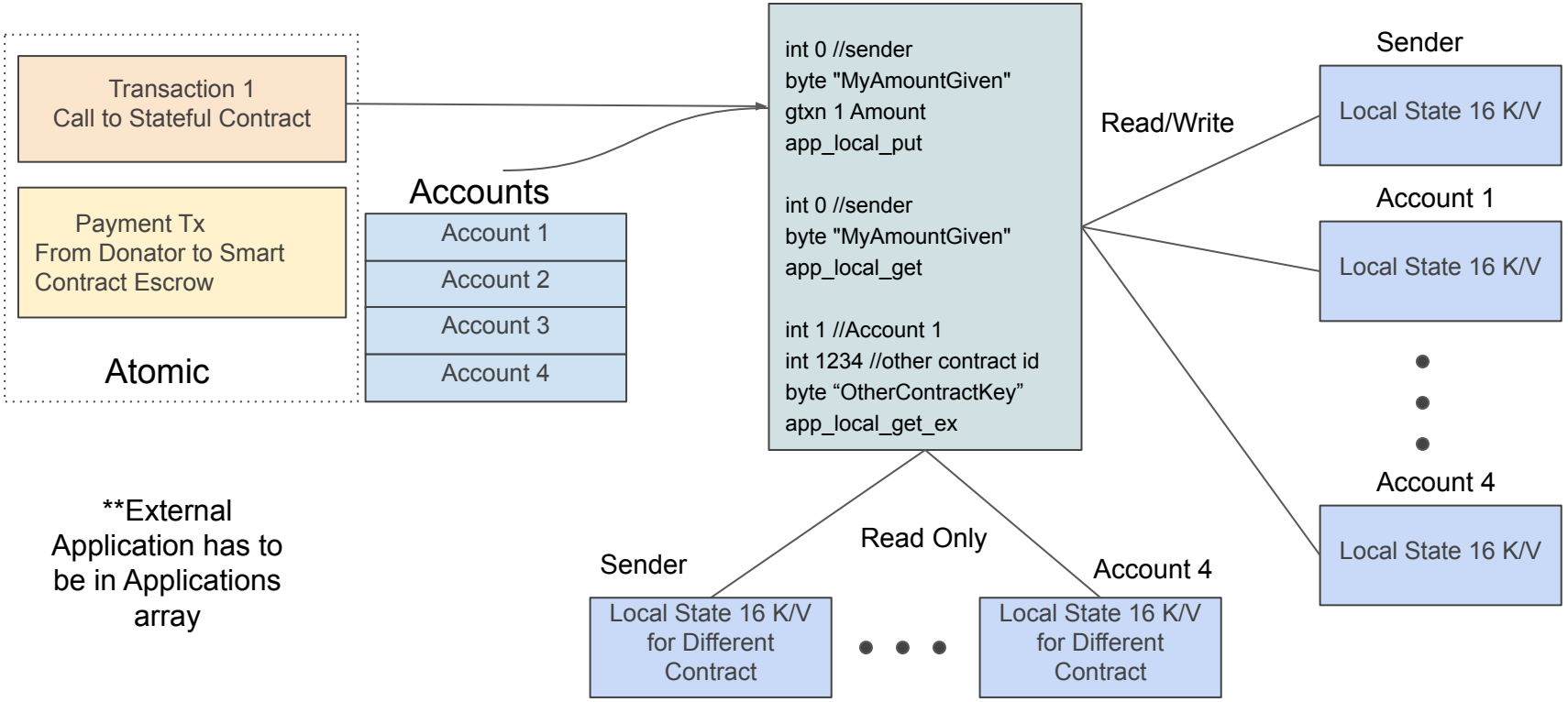
**Asset and Account have to be in Assets and Accounts array

Read Global State From Another Contract



**External Application has to be in Applications array

Reading and Writing Local State



**External Application has to be in Applications array

Loops and Subroutines

- **Constraint -opcode budget**
  - 700 opcode per call
  - Can be pooled with up to 16 atomic application transactions
- **Constraint - file size**
  - 2k - 8k for total for both programs
  - Upping pages increase minimum balance

```
main:
int 0 // initialize loop
loop:
callsub cal_mul
int 1
+ // increment by 1
dup
txn ApplicationArgs 0
btoi // it must be an integer
int 1
-
< // loop until app argument -1
bnz loop
```

```
cal_mul:
load 0
int 1
-
load 1
*
store 1
load 0
int 1
-
store 0
retsub
```

Algorand

Looping and Subroutine Demo

Inner Transactions

- Up to 16 inner transactions
- Smart contract pays fees
  - Fees eligible for pooling
  - Fees are default min
- Shows up as inner transactions within application transaction
- Recipient must be in the accounts array
- Accounts can be rekeyed to smart contract giving the spending authority to the contract but account must be in accounts array to spend from it
- Supports all asset transactions
- Supports payment transactions

```
handle_noop:
txn ApplicationArgs 0
byte "payme"
==
assert
itxn_begin
int pay
itxn_field TypeEnum
int 5000
itxn_field Amount
txn Sender
itxn_field Receiver
itxn_submit
```

Algorand

PyTeal

PyTEAL -Python Library

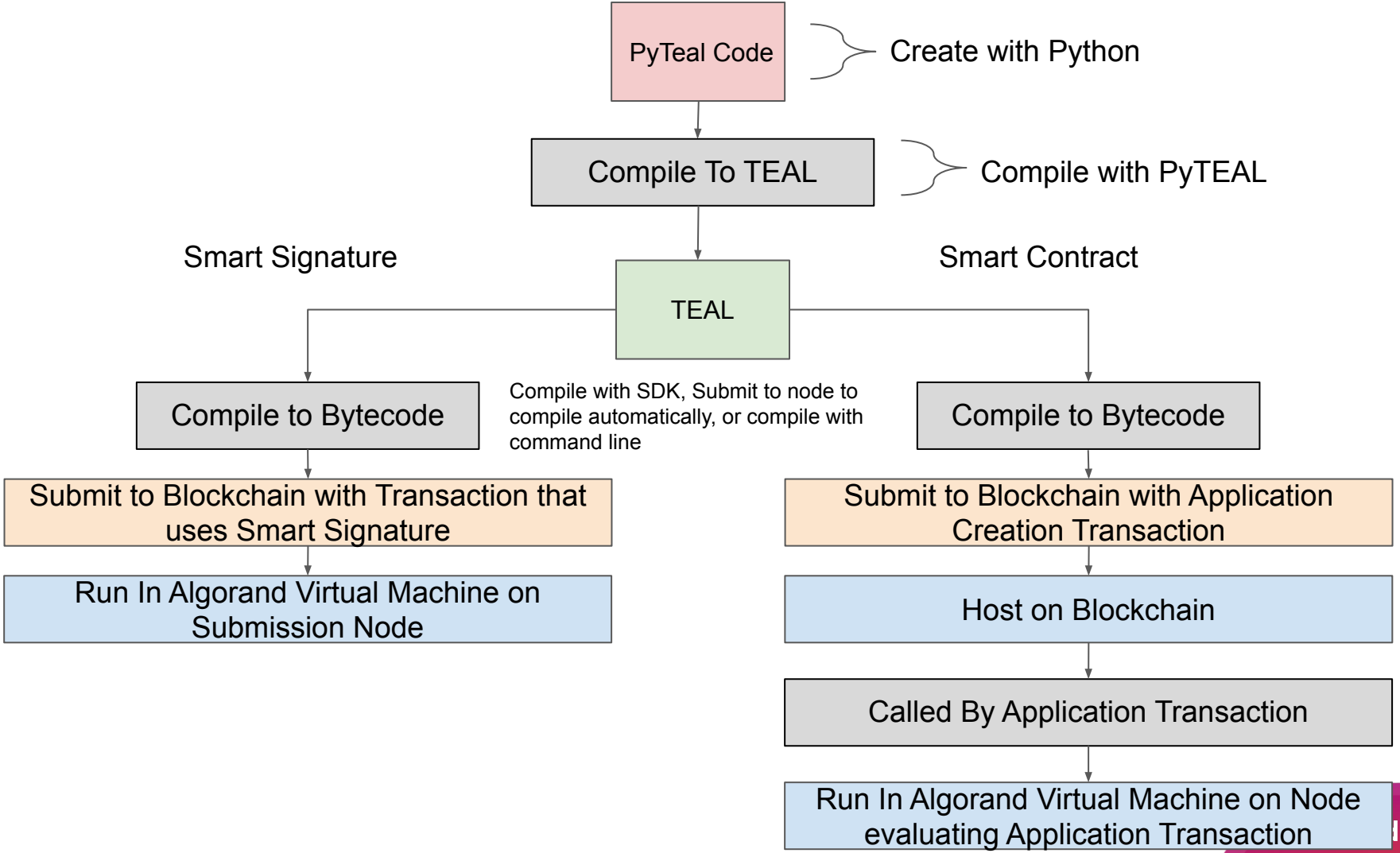
```
def bank_for_account(receiver):
    is_payment = Txn.type_enum() == TxnType.Payment
    is_single_tx = Global.group_size() == Int(1)
    is_correct_receiver = Txn.receiver() == Addr(receiver)

    return And(
        is_payment,
        is_single_tx,
        is_correct_receiver
    )
if __name__ == "__main__":
    program =
bank_for_account("ZZAF5ARA4MEC5PVDOP64JM5O5MQST63Q2KOY2FLYFL
XXD3PFSNJJBYAFZM")
    print(compileTeal(program, Mode.Signature))
```

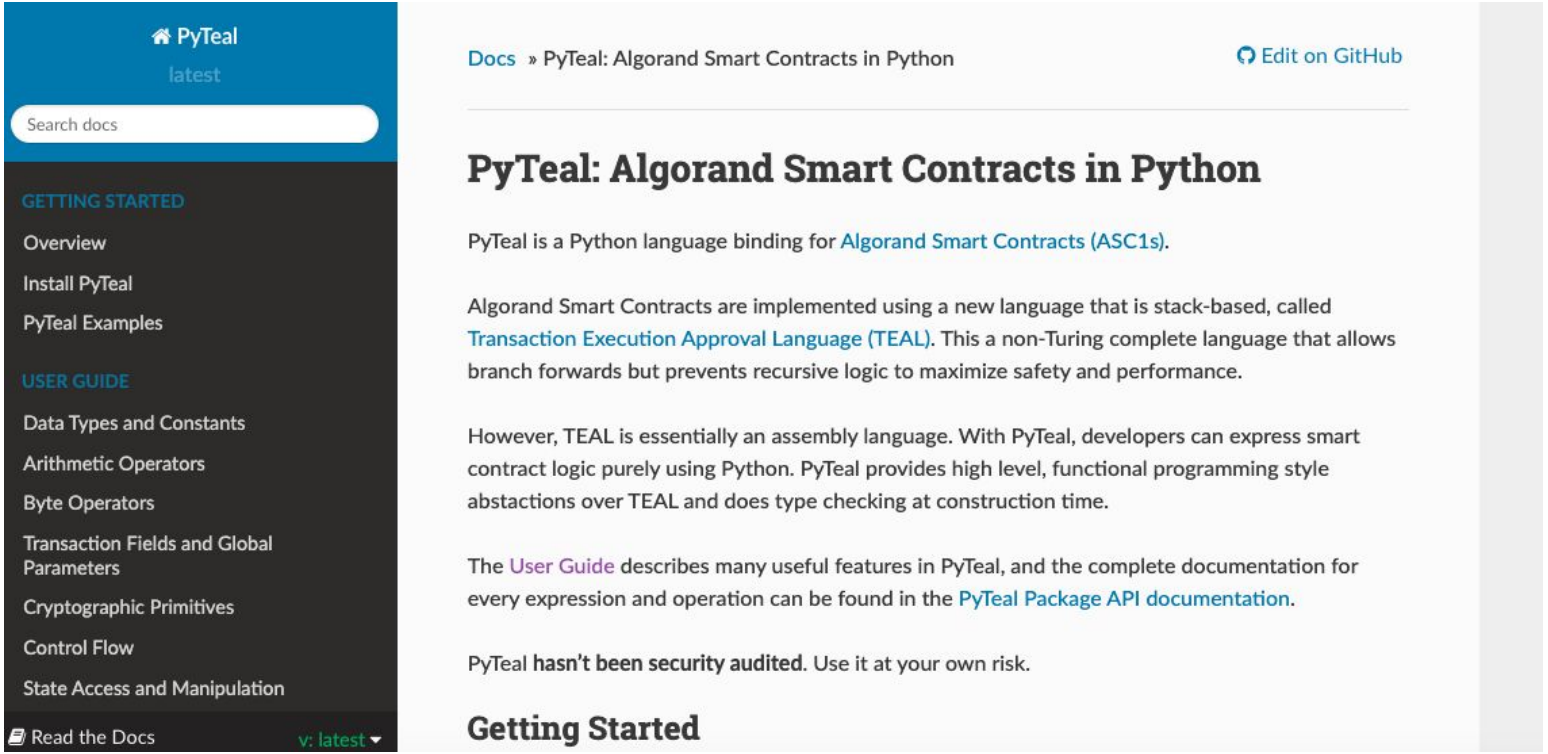
Returns

```
#pragma version 2
txn TypeEnum
int pay
==
global GroupSize
int 1
==
&&
txn Receiver
addr
ZZAF5ARA4MEC5PVDOP64JM5O5MQST63Q2KOY2FLYFLXXD3P
FSNJJBYAFZM
==
&&
```

Algorand


PyTeal Code
Create with Python
Compile To TEAL
Compile with PyTEAL
Smart Signature
Smart Contract
TEAL
Compile to Bytecode
Compile with SDK, Submit to node to compile automatically, or compile with command line
Compile to Bytecode
Submit to Blockchain with Transaction that uses Smart Signature
Run In Algorand Virtual Machine on Submission Node
Submit to Blockchain with Application Creation Transaction
Host on Blockchain
Called By Application Transaction
Run In Algorand Virtual Machine on Node evaluating Application Transaction

PyTeal Documentation

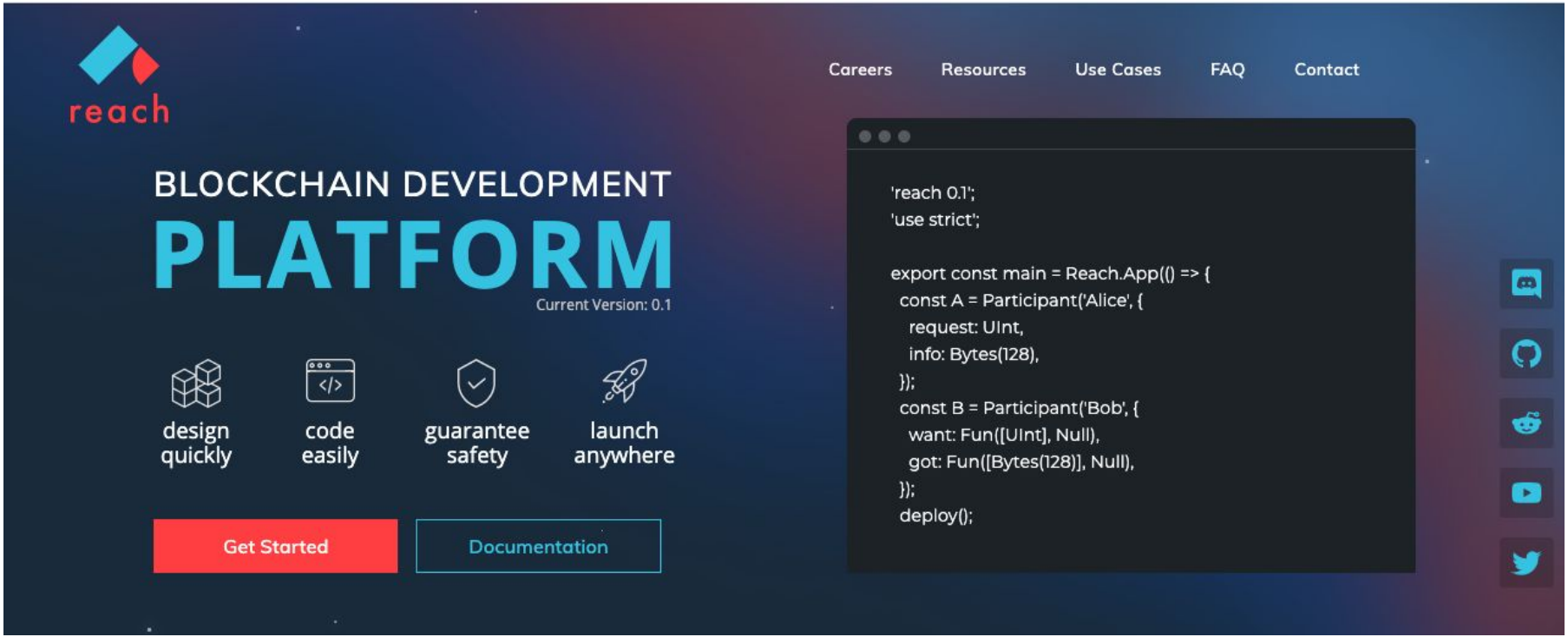


[Documentation]

Algorand

PyTeal Demo

Reach - Reach.sh



Link To Presentation

- https://github.com/algorand/smart-contracts/tree/master/devrel/innovate

Resources

- **Discord:** https://discord.com/invite/84AActu3at
- Developer Portal (Documentation and Tutorials): https://developer.algorand.org/
- Forum: https://forum.algorand.org/
- GitHub: https://github.com/algorand
- OFFICE HOURS sign up: https://www.algorand.com/developers