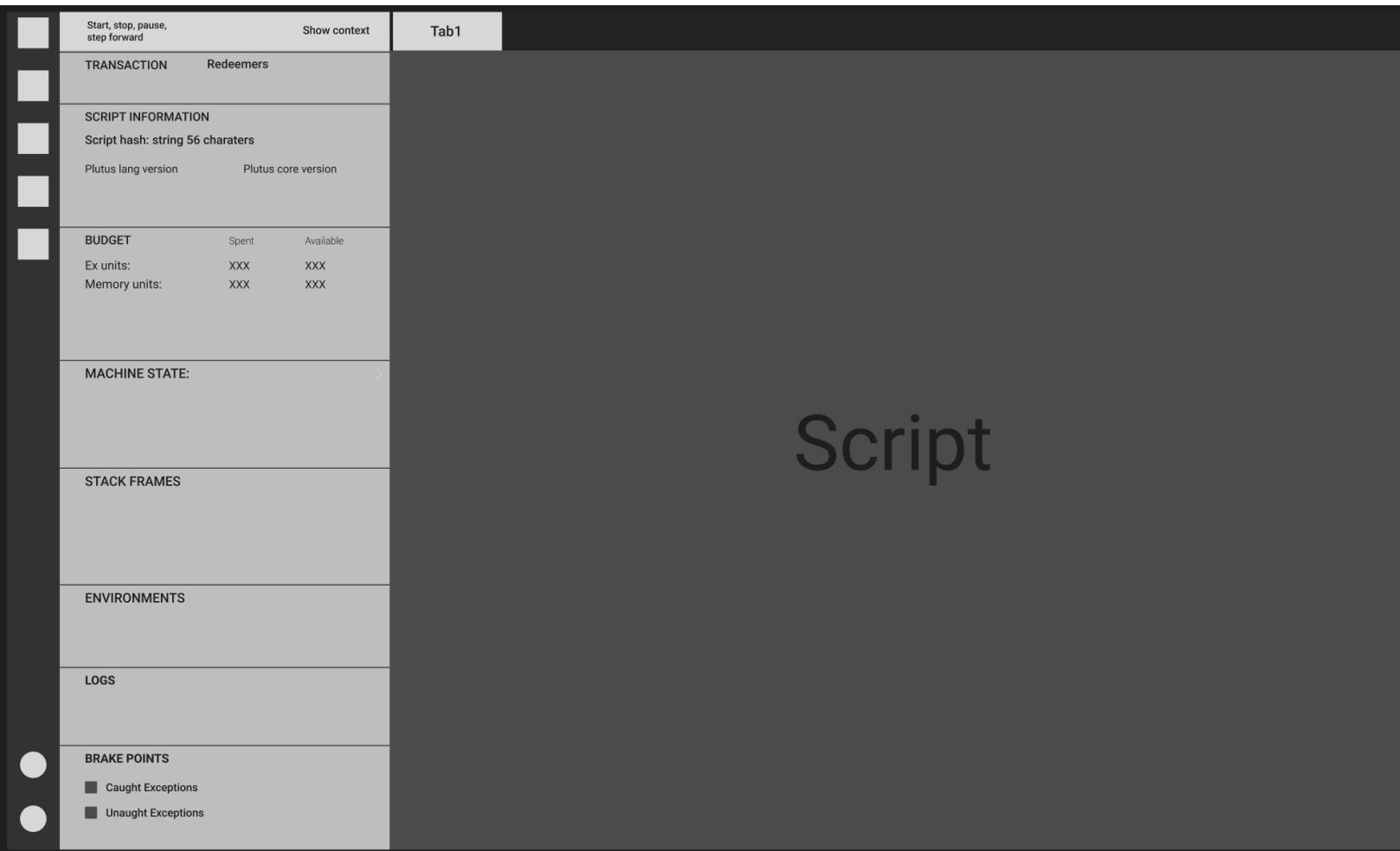


NOTE: you can always zoom it to see details.

Wireframe:

Initial idea of UI based in VS Code design



Mockups:
No active debugging example:

The screenshot shows a debugger interface with the following components:

- Top Bar:** Includes icons for file operations (New, Open, Save, etc.), "Show context" toggle, and tabs for "Untitled-1", "Untitled-2", and "Program".
- Left Sidebar:** Contains sections for "TRANSACTION" (Redeemers), "SCRIPT INFORMATION" (Script hash: 0645213af4312bc1223ef, Plutus lang version: 1.03, Plutus core version: 1.54), "BUDGET" (Ex units: 0, Memory units: 0), "MACHINE STATE", "STACK FRAMES", "ENVIRONMENTS", "LOGS", and "BRAKE POINTS".
- Right Editor Area:** Displays a code editor with the following content:

```
1  {
2   term_type: Case,
3   term_name: "case1",
4   constr: {
5     term_type: Var,
6     term_name: "var_Constr",
7     variable_name: "someConstructor",
8   },
9   branches: [
10    {
11      term_type: Lambda,
12      term_name: "branch1",
13      parameter_name: "x",
14      body: {
15        term_type: Var,
16        term_name: "var_x",
17        variable_name: "x",
18      },
19    },
20    {
21      term_type: Lambda,
22      term_name: "branch2",
23      parameter_name: "y",
24      body: {
25        term_type: Apply,
26        term_name: "apply_negate",
27        function: (...),
28        argument: {
29          term_type: Var,
30          term_name: "var_y",
31          variable_name: "y"
32        },
33      },
34    },
35  },
36  (...)
```

- Status Bar:** Shows "main" tab, file statistics (0:11, 0:0), and system information (Ln 17, Col 3, Spaces: 2, UTF-8, LF, UPLC).

Left Sidebar

1. Debug Controls
 - Flow control buttons (Play, Step, Stop)
 - "Show context" toggle
2. Transaction Settings
 - Redeemer selector for script execution
3. Script Info
 - Hash
 - Plutus versions: lang and core
4. Resource Monitor
 - Ex units usage
 - Memory units usage
5. Machine State
 - Current execution state display
6. Stack Frames
 - Execution context list

- 7. Environments
 - Variable values list
- 8. Logs
 - "trace" instruction outputs
- 9. Breakpoints
 - Debug stop points list

Main Area (Right)

- UPLC code display with syntax highlighting
- Line numbers and current execution point
- Tab-based file navigation

DE-UPLC logo on in the left area.

Example for active debug session

```

TRANSACTION > Redeemers

SCRIPT INFORMATION
Script hash: 0645213af4312bc1223ef
Plutus lang version: 1.03 Plutus core version: 1.54

BUDGET
Spent Available
Ex units: 52 456, 45 645 458, 56
Memory units: 52 456, 45 645 458, 56

MACHINE STATE:
Return: Constant(Integer(42))

STACK FRAMES
FrameForce
FrameAwaitArg
  function: Value

ENVIRONMENTS
Constant: Integer
Delay
  body: Term("Name")

LOGS
String1
String2

BRAKE POINTS
Line: 2
Line: 5

```

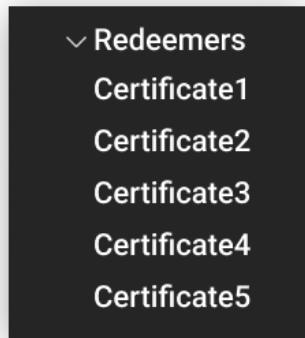
```

1  {
2   term_type: Case,
3   term_name: "case1",
4   constr: (
5     term_type: Var,
6     term_name: "var_constr",
7     variable_name: "someConstructor",
8   },
9   branches: [
10    {
11      term_type: Lambda,
12      term_name: "branch1",
13      parameter_name: "x",
14      body: {
15        term_type: Var,
16        term_name: "var_x",
17        variable_name: "x",
18      },
19    },
20    {
21      term_type: Lambda,
22      term_name: "branch2",
23      parameter_name: "y",
24      body: {
25        term_type: Apply,
26        term_name: "apply_negate",
27        function: (
28          term_type: Builtin,
29          term_name: "builtin_negate",
30          function_name: "negate"
31        ),
32        argument: {
33          term_type: Var,
34          term_name: "var_y",
35          variable_name: "y"
36        },
37        ...
38      }
39    }
40  }
41}

```

Ln 17, Col 3 Spaces: 2 UTF-8 LF UPLC

Example of Redeemers selector:



Next UI mockups are representation of most of components that we expect to see on left sidebar:

Note: this button with arrow means that component has a data that will be opened by pressing it. Data will be shown in separate tab in json-like representation.

```
graph TD; Value[Value: Constant: >]; Value --- Delay[Delay: body: Term("Name") > environments: >]; Value --- Lambda[Lambda: parameter_name: 2364 body: Term(1254) > environments: ... >]; Value --- Builtin[Builtin: fun: DefaultFunction runtime: BuiltinRuntime]; Value --- Constr[Constr: tag: 5654 fields >]
```

FrameForce:

FrameAwaitArg:

function: Value [🔗](#)

FrameAwaitFunTerm:

environments: ... [🔗](#)

argument: Term<Name> [🔗](#)

FrameAwaitFunValue:

argument: Value [🔗](#)

FrameForce

environments: ... [🔗](#)

tag: 45668

✓ fields_to_resolve:

Term<Name> [🔗](#)

resolved_fields: Vec<Value> [🔗](#)

Constr:

environments: ... [🔗](#)

✓ branches:

Term<Name> [🔗](#)

NoFrame

DefaultFunction:

- AddInteger
- SubtractInteger
- MultiplyInteger
- DivideInteger
- QuotientInteger
- RemainderInteger
- ModInteger
- EqualsInteger
- LessThanInteger
- LessThanEqualsInteger

Type:

- Bool
- Integer
- String
- ByteString
- Unit
- List (Data)
- Pair (Bool, Intenger)
- Data
- Bls12_381G1Element
- Bls12_381G2Element
- Bls12_381MIResult

```
> List  
> List  
  Data  
  
> Pair  
> Pair  
  Bool  
  Intenger  
  Bool
```

```
> List  
  ∵ List  
  
> Pair  
  ∵ Pair  
    Bool
```

```
  ∵ List  
  ∵ Pair
```

Constant:

[Integer \(42\)](#)

[ByteString \(0645213af4312bc1223ef\)](#)

[String \("kuggjkjkhkj"\)](#)

[UnsignedInteger \(67\)](#)

[Bool \(true\)](#)

[ProtoList \(Bool, Vec <Constant>\)](#) ▾

[ProtoPair \(String, Type, Constant, Constant\)](#) ▾

[Data \(PlutusData\)](#) ▾

[Bls12_381G1Element](#) ▾

[Bls12_381G1Element](#) ▾

[Bls12_381MIResult](#) ▾

Term:

Var (Name)

Delay (Term<Name>) ▾

Lambda:

parameter_name: fhh15jk

body: Term<Name> ▾

Apply:

function: Term<Name> ▾

argument: Term<Name> ▾

Constant (Constant)

Force (Term<Name>) ▾

Error

Builtin (DefaultFunction)

Constr:

tag: usize

 └ fields:

 Term<Name> ▾

Case:

constr: Term<Name> ▾

 └ branches:

 Term<Name> ▾

Next UI mockups are representation of most of components that we expect to see on a tab with json-like representation:

Terms:

Var

```
{  
  term_type: Var,  
  term_name: "unique_term_name",  
  variable_name: "variable_name"  
}
```

Force

```
{  
  term_type: Force,  
  term_name: "unique_term_name",  
  term: /* Nested Term */  
}
```

Delay

```
{  
  term_type: Delay,  
  term_name: "unique_term_name",  
  term: /* Nested Term */  
}
```

Error

```
{  
  term_type: Error,  
  term_name: "unique_term_name",  
}
```

Lambda

```
{  
  term_type: Lambda,  
  term_name: "unique_term_name",  
  parameter_name: "parameter_name",  
  body: /* Nested Term */  
}
```

Builtin

```
{  
  term_type: Builtin,  
  term_name: "unique_term_name",  
  function_name: "function_name"  
}
```

Apply

```
{  
    term_type: Apply,  
    term_name: "unique_term_name",  
    function: { /* Nested Term */ }  
    argument: { /* Nested Term */ }  
}
```

Constr

```
{  
    term_type: Constr,  
    term_name: "unique_term_name",  
    tag: integer,  
    fields: [ /* Array of Term */ ]  
}
```

Constant

```
{  
    term_type: Constant,  
    term_name: "unique_term_name",  
    value: /* Constant value */  
}
```

Case

```
{  
    term_type: Case,  
    term_name: "unique_term_name",  
    constr: { /* Constructor Term */ },  
    branches: [ /* Array of Branch Term */ ]  
}
```

Constants

Integer

```
{  
    constant_type: Integer,  
    value: "string_representation_of_bigint"  
}
```

Bool

```
{  
    constant_type: Bool,  
    value: "true_or_false"  
}
```

ByteString

```
{  
    constant_type: ByteString,  
    value: "hex_encoded_string"  
}
```

ProtoList

```
{  
    constant_type: ProtoList,  
    element_type: { /* Type */ },  
    elements: [ /* Array of Constants */ ]  
}
```

String

```
{  
    constant_type: String,  
    value: "your_string_valuet"  
}
```

ProtoPair

```
{  
    constant_type: ProtoPair,  
    left_type: { /* Type */ },  
    right_type: { /* Type */ },  
    left_value: { /* Constant */ },  
    right_value: { /* Constant */ }  
}
```

Unit

```
{  
    constant_type: Unit,  
}
```

Data

```
{  
    constant_type: Data,  
    value: { /* PlutusData */ }  
}
```

BLS Constans:

Bls12_381G1Element

```
{  
    constant_type: Bls12_381G1Element,  
    value: { /* blst_p1 */ }  
}
```

blst_p1

```
{  
    x: { /* blst_fp */ },  
    y: { /* blst_fp */ },  
    z: { /* blst_fp */ }  
}
```

Bls12_381G2Element

```
{  
    constant_type: Bls12_381G2Element,  
    value: { /* blst_p2 */ }  
}
```

blst_fp

```
{  
    l: [ 1, 3 ,66 , 99, 33, 123456, ]  
}
```

Bls12_381MIResult

```
{  
    constant_type: Bls12_381MIResult,  
    value: { /* blst_fp12 */ }  
}
```

blst_fp2

```
{  
    fp: [  
        { /* blst_fp */ },  
        { /* blst_fp */ }  
    ]  
}
```

blst_p2

```
{  
    x: { /* blst_fp2 */ },  
    y: { /* blst_fp2 */ },  
    z: { /* blst_fp2 */ }  
}
```

blst_fp6

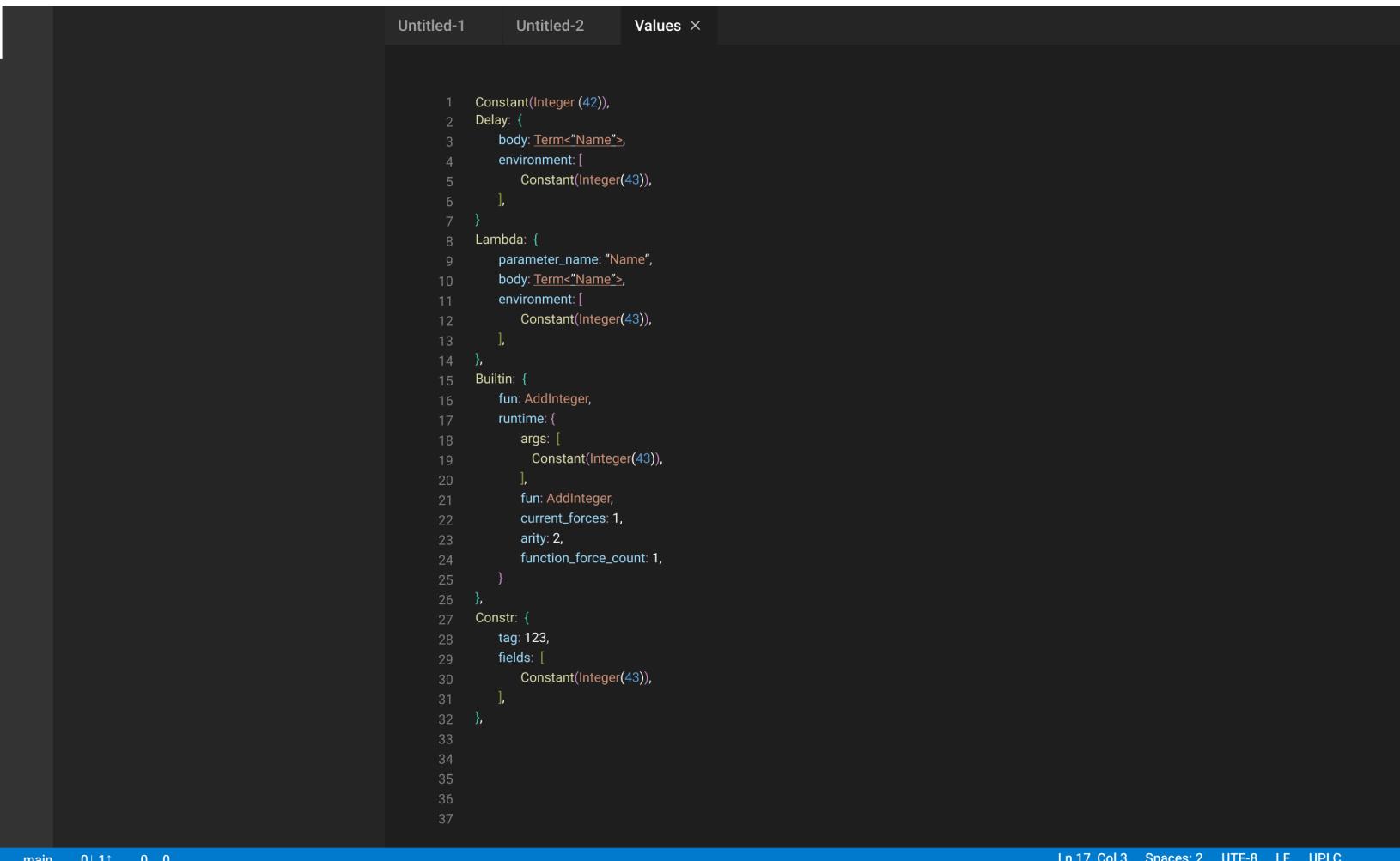
```
{  
    fp: [  
        { /* blst_fp2 */ },  
        { /* blst_fp2 */ },  
        { /* blst_fp2 */ }  
    ]  
}
```

Example tab with constant:

Untitled-1 Untitled-2 Constant ×

```
1  {
2    constant_type: ProtoList,
3    element_type: {
4      type: Integer
5    },
6    elements: [
7      {
8        constant_type: Integer,
9        value: "1"
10      },
11      {
12        constant_type: Integer,
13        value: "2"
14      },
15      {
16        constant_type: Integer,
17        value: "3"
18      }
19    ]
20  }
```

Example tab with Value:



```
Untitled-1 Untitled-2 Values ×

1 Constant(Integer(42)),
2 Delay: {
3     body: Term<"Name">,
4     environment: [
5         Constant(Integer(43)),
6     ],
7 }
8 Lambda: {
9     parameter_name: "Name",
10    body: Term<"Name">,
11    environment: [
12        Constant(Integer(43)),
13    ],
14 },
15 Builtin: {
16     fun: AddInteger,
17     runtime: {
18         args: [
19             Constant(Integer(43)),
20         ],
21         fun: AddInteger,
22         current_forces: 1,
23         arity: 2,
24         function_force_count: 1,
25     },
26 },
27 Constr: {
28     tag: 123,
29     fields: [
30         Constant(Integer(43)),
31     ],
32 },
33
34
35
36
37
```

main 0 1 1 0 0 Ln 17, Col 3 Spaces: 2 UTF-8 LF UPLC

Example of script context:

```
Script context
},
inputs: [
{
out_ref: {
tx_id: 1ab2c3d4e5f67890a1b2c3d4e5f67890a1b2c3d4e5f67890a1b2c3d4e5f67890
output_index: 0
},
resolved: {
address: addr1qxy2k8hrkgeqmnapgx7jvg9rc0gkrmgstx4yjk9k9kxvnq7m8a9fu9x7dzkg9amgvf2v5hz42zc7glf3kz7df4m4mass8nafx,
value: {
coin: 1000000
assets: {
b0d07d45fe9514f80213f4020e5a61241458be626841cde717cb38a76e7574686572756d: 50
c0ffd7cb14cf0cba9c862d0aa6d0d96cece28c9e5bf7e7f6a1d58d426974636f696e: 100
},
},
datum_hash: ab12cd34ef56gh78ij90kl12mn34op56qr78st90uv12wx34yz56ab12cd34ef56
},
},
outputs: [
{
address: addr1qx8phkx6acpnf7275k6vs3fkpxk7zsxw2uwqafy788txxm8a9fu9x7dzkg9amgvf2v5hz42zc7glf3kz7df4m4masfjx5p7,
value: {
coin: 500000
assets: {
b0d07d45fe9514f80213f4020e5a61241458be626841cde717cb38a76e7574686572756d: 25
},
},
datum_hash: ff34de5678ab90cd12ef34gh56ij78kl90mn12op34qr56st78uv90wx12yz34ab56
},
],
fee: {
coin: 180000
},
mint: {
tokens: {
d9312da562da182b02322fd8acb536f37eb9d29fba7c49dc12784d0f446e77546f6b656e: 1000
}
},
certificates: [
{
type: StakeRegistration,
stake_credential: e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2
},
],
withdrawals: [
[
addr1qxy2k8hrkgeqmnapgx7jvg9rc0gkrmgstx4yjk9k9kxvnq7m8a9fu9x7dzkg9amgvf2v5hz42zc7glf3kz7df4m4mass8nafx,
1000000
]
],
valid_range: {
lower_bound: 41000000,
upper_bound: 42000000
},
signatories: [
a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2,
b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3
],
data: [
[
d1e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9c0d1e2,
{
constructor: 0,
fields: [
{int: 42},
{bytes: 48656c6c6f20576f726c6421}
]
}
],
],
redeemers: {
Spend: 0
purpose: Spend,
index: 0,
data: {
constructor: 0,
fields: [],
{int: 1}
],
ex_units: {
mem: 1000000,
steps: 700000
}
},
id: 3abc4def5678901234567890abcdef1234567890abcdef1234567890abcdef12
}
```