

Audit

By OCamlPro

• September 7, 2021

Table of Major and Critical Issues

Critical issue: Constructor for Data (fake)	29
Critical issue: Constructor for Index (fake)	36
Critical issue: Constructor for IndexBasis (fake)	39
Critical issue: Constructor for Manager (fake)	44
Critical issue: Constructor for NftRoot (fake)	50

Contents

1	Only for Auditors	6
1.1	To edit this documents	6
1.2	General Auditing Rules	6
2	Introduction	8
3	Overview	9
4	Library Modules	10
4.1	Module "Constants.sol"	11
4.1.1	Pragmas	11
4.1.2	Contract Definitions	11
4.2	Module "Errors.sol"	12
4.2.1	Pragmas	12
4.2.2	Contract Definitions	12
4.3	Module "true_nft_audit.sol"	13
4.3.1	Imports	13
5	Interface Modules	14
5.1	Module "IData.sol"	15
5.1.1	Pragmas	15
5.1.2	Contract Definitions	15
5.2	Module "IIndex.sol"	16
5.2.1	Pragmas	16
5.2.2	Contract Definitions	16
5.3	Module "IIndexBasis.sol"	17
5.3.1	Pragmas	17
5.3.2	Contract Definitions	17
6	Contract Modules	18
6.1	Module "Data.sol"	19
6.1.1	Pragmas	19
6.1.2	Imports	19
6.1.3	Contract Definitions	19

6.2	Module "DataResolver.sol"	20
6.2.1	Pragmas	20
6.2.2	Imports	20
6.2.3	Contract Definitions	20
6.3	Module "Index.sol"	21
6.3.1	Pragmas	21
6.3.2	Imports	21
6.3.3	Contract Definitions	21
6.4	Module "IndexBasis.sol"	22
6.4.1	Pragmas	22
6.4.2	Imports	22
6.4.3	Contract Definitions	22
6.5	Module "IndexResolver.sol"	23
6.5.1	Pragmas	23
6.5.2	Imports	23
6.5.3	Contract Definitions	23
6.6	Module "Manager.sol"	24
6.6.1	Pragmas	24
6.6.2	Imports	24
6.6.3	Contract Definitions	24
6.7	Module "NftRoot.sol"	25
6.7.1	Pragmas	25
6.7.2	Imports	25
6.7.3	Contract Definitions	25
7	Contract Data	26
7.1	Overview	26
7.2	Contract Inheritance	26
7.3	Static Variable Definitions	26
7.4	Variable Definitions	28
7.5	Constructor Definitions	29
7.5.1	Constructor	29
7.6	Public Method Definitions	30
7.6.1	Function destruct	30
7.6.2	Function getInfo	30
7.6.3	Function getOwner	31
7.7	Internal Method Definitions	31
7.7.1	Function deployIndex	31
8	Contract DataResolver	32
8.1	Overview	32
8.2	Variable Definitions	32
8.3	Public Method Definitions	33
8.3.1	Function resolveCodeHashData	33
8.3.2	Function resolveData	33
8.4	Internal Method Definitions	33

8.4.1	Function <code>_buildDataCode</code>	33
8.4.2	Function <code>_buildDataState</code>	33
9	Contract Index	35
9.1	Overview	35
9.2	Contract Inheritance	35
9.3	Static Variable Definitions	35
9.4	Variable Definitions	36
9.5	Constructor Definitions	36
9.5.1	Constructor	36
9.6	Public Method Definitions	37
9.6.1	Function <code>destruct</code>	37
9.6.2	Function <code>getInfo</code>	37
10	Contract IndexBasis	38
10.1	Overview	38
10.2	Static Variable Definitions	38
10.3	Modifier Definitions	39
10.3.1	Modifier <code>onlyRoot</code>	39
10.4	Constructor Definitions	39
10.4.1	Constructor	39
10.5	Public Method Definitions	39
10.5.1	Function <code>destruct</code>	39
10.5.2	Function <code>getInfo</code>	39
11	Contract IndexResolver	40
11.1	Overview	40
11.2	Variable Definitions	40
11.3	Public Method Definitions	41
11.3.1	Function <code>resolveCodeHashIndex</code>	41
11.3.2	Function <code>resolveIndex</code>	41
11.4	Internal Method Definitions	41
11.4.1	Function <code>_buildIndexCode</code>	41
11.4.2	Function <code>_buildIndexState</code>	41
12	Contract Manager	43
12.1	Overview	43
12.2	Variable Definitions	43
12.3	Constructor Definitions	44
12.3.1	Constructor	44
12.4	Public Method Definitions	44
12.4.1	Function <code>deployRoot</code>	44
12.5	Internal Method Definitions	44
12.5.1	Function <code>_buildNftRootState</code>	44

13 Contract NftRoot	46
13.1 Overview	46
13.2 Contract Inheritance	46
13.3 Static Variable Definitions	47
13.4 Variable Definitions	49
13.5 Modifier Definitions	50
13.5.1 Modifier onlyOwner	50
13.6 Constructor Definitions	50
13.6.1 Constructor	50
13.7 Public Method Definitions	51
13.7.1 Function burn	51
13.7.2 Function deployBasis	51
13.7.3 Function destructBasis	52
13.7.4 Function getInfo	52
13.7.5 Function mintNft	52
13.7.6 Function setPrice	53

Chapter 1

Only for Auditors

1.1 To edit this documents

In the report.tex file, choose:

- `\soldraftfalse` to remove draft mode (watermarks, advises)
- `\solmodulestrue` to display modules by chapter instead of contracts
- `\soltablestrue` to display tables for parameters and returns
- `\solissuesfalse` to remove the table of issues

Issues can be entered with:

- `\issueCritical{title}{text}`
- `\issueMajor{title}{text}`
- `\issueMinor{title}{text}`

1.2 General Auditing Rules

- Check that types have the correct integer types (Pubkey : uint256, Amount: uint128, Time: uint64).
- Naming conventions: constants should for example be all uppercase, static variables should start with a prefix like `s_`, globals should start with a prefix like `g_` or `m_`, internal functions should start with a prefix `_`.
- Numbers should not appear in source, but be defined as constants.
- In constant definitions, verify that 2 consecutive errors have not the same error (common copy-paste error)

- Constants for amounts should be expressed in `ton` to prevent too many zeroes.
- Modifiers with `tvm.accept` must always check the source of the message
- Constructors with arguments must always check the source of the message to prevent anybody from calling the constructor and set variables instead of the real owner
- Failures should never happen after `tvm.accept` (such as `require`, division by zero, overflows, etc.)
- Most arguments should be protected by a `require`
- Before sending a message, the function should check that it has enough gas (to prevent a partial failure during the message sending phase)
- `tvm.accept` should only be called after verifying that the sender of the message is the contracts' owner

Chapter 2

Introduction

Chapter 3

Overview

Chapter 4

Library Modules

4.1 Module "Constants.sol"

4.1.1 Pragas

ton	-solidity >= 0.43.0	
-----	---------------------	--

4.1.2 Contract Definitions

- Constants

4.2 Module "Errors.sol"

4.2.1 Pragas

ton	-solidity >= 0.43.0	
-----	---------------------	--

4.2.2 Contract Definitions

- Errors

4.3 Module "true_nft_audit.sol"

4.3.1 Imports

../share/surfer/src/NftRoot.sol	
../share/surfer/src/Manager.sol	

Chapter 5

Interface Modules

5.1 Module "IData.sol"

5.1.1 Pragas

ton	-solidity >= 0.43.0	
-----	---------------------	--

5.1.2 Contract Definitions

- IData

5.2 Module "IIndex.sol"

5.2.1 Pragas

ton	-solidity >= 0.43.0	
-----	---------------------	--

5.2.2 Contract Definitions

- IIndex

5.3 Module "IIndexBasis.sol"

5.3.1 Pragas

ton	-solidity >= 0.43.0	
-----	---------------------	--

5.3.2 Contract Definitions

- IIndexBasis

Chapter 6

Contract Modules

6.1 Module "Data.sol"

6.1.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

6.1.2 Imports

./resolvers/IndexResolver.sol	
./interfaces/IData.sol	
./libraries/Constants.sol	
./libraries/Errors.sol	

6.1.3 Contract Definitions

- Data

6.2 Module "DataResolver.sol"

6.2.1 Pragas

ton	-solidity >= 0.43.0	
AbiHeader	expire	
AbiHeader	time	

6.2.2 Imports

../Data.sol	
-------------	--

6.2.3 Contract Definitions

- DataResolver

6.3 Module "Index.sol"

6.3.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

6.3.2 Imports

./interfaces/IIndex.sol	
./libraries/Errors.sol	

6.3.3 Contract Definitions

- Index

6.4 Module "IndexBasis.sol"

6.4.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

6.4.2 Imports

./libraries/Errors.sol	
------------------------	--

6.4.3 Contract Definitions

- IndexBasis

6.5 Module "IndexResolver.sol"

6.5.1 Pragas

ton	-solidity >= 0.43.0	
AbiHeader	expire	
AbiHeader	time	

6.5.2 Imports

../Index.sol	
--------------	--

6.5.3 Contract Definitions

- IndexResolver

6.6 Module "Manager.sol"

6.6.1 Pragas

ton	-solidity >= 0.43.0	
AbiHeader	expire	
AbiHeader	time	

6.6.2 Imports

./NftRoot.sol	
./libraries/Constants.sol	
./libraries/Errors.sol	

6.6.3 Contract Definitions

- Manager

6.7 Module "NftRoot.sol"

6.7.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

6.7.2 Imports

./resolvers/IndexResolver.sol	
./resolvers/DataResolver.sol	
./IndexBasis.sol	
./interfaces/IIndexBasis.sol	
./libraries/Constants.sol	
./libraries/Errors.sol	

6.7.3 Contract Definitions

- NftRoot

Chapter 7

Contract Data

Contents

7.1	Overview	26
7.2	Contract Inheritance	26
7.3	Static Variable Definitions	26
7.4	Variable Definitions	28
7.5	Constructor Definitions	29
7.5.1	Constructor	29
7.6	Public Method Definitions	30
7.6.1	Function destruct	30
7.6.2	Function getInfo	30
7.6.3	Function getOwner	31
7.7	Internal Method Definitions	31
7.7.1	Function deployIndex	31

7.1 Overview

In file `Data.sol`

7.2 Contract Inheritance

IData	
IndexResolver	

7.3 Static Variable Definitions

uint256	_id	
---------	-----	--

```
17  uint256 static _id;
```

7.4 Variable Definitions

address	_addrRoot	
		used in @9.Data.destruct
		used in @9.Data.destruct
		used in @9.Data.deployIndex
		used in @9.Data.deployIndex
		used in @9.Data.deployIndex
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
address	_addrOwner	
		used in @9.Data.getOwner
		used in @9.Data.destruct
		used in @9.Data.destruct
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
address	_addrAuthor	
		used in @9.Data.getInfo
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
string	_name	
		used in @9.Data.getInfo
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
string	_description	
		used in @9.Data.getInfo
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
string	_tokenCode	
		used in @9.Data.getInfo
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
uint64	_creationDate	
		used in @9.Data.getInfo
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
string	_comment	
		used in @9.Data.getInfo
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
mapping (uint128 => bytes)	_content	
		used in @9.Data.getInfo
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor

```
14     address _addrRoot;
15     address _addrOwner;
16     address _addrAuthor;
17
18     string _name;
19
20     string _description;
21
22     string _tokenCode;
23
24     uint64 _creationDate;
25
26     string _comment;
27
28     mapping(uint128 => bytes) _content;
```

7.5 Constructor Definitions

7.5.1 Constructor

Critical issue: Constructor for Data (fake)

lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
ipsum lorem ipsum lorem ipsum

- TODO

```

28 constructor(
29     address addrOwner,
30     TvmCell codeIndex,
31     address addrAuthor,
32     string name,
33     string description,
34     string tokenCode,
35     uint64 creationDate,
36     string comment,
37     uint128 index,
38     bytes part
39 ) public {
40     optional(TvmCell) optSalt = tvvm.codeSalt(tvm.code());
41     require(optSalt.hasValue(), Errors.ERROR_EMPTY_SALT);
42     (address addrRoot) = optSalt
43         .get()
44         .toSlice()
45         .decode(address);
46     require(msg.sender == addrRoot, Errors.
        ERROR_MESSAGE_SENDER_IS_NOT_ROOT);

```

```

47     require(msg.value >= Constants.MIN_FOR_DEPLOY);
48     _addrRoot = addrRoot;
49     _addrOwner = addrOwner;
50     _addrAuthor = addrAuthor;
51     _name = name;
52     _description = description;
53     _tokenCode = tokenCode;
54     _creationDate = creationDate;
55     _comment = comment;
56     _codeIndex = codeIndex;
57
58     _content[index] = part;
59
60     deployIndex(addrOwner);
61 }

```

7.6 Public Method Definitions

7.6.1 Function destruct

- TODO

```

77     function destruct(address recipient) public {
78         require(msg.sender == _addrRoot, Errors.
79             ERROR_MESSAGE_SENDER_IS_NOT_ROOT);
80
81         address oldIndexOwner = resolveIndex(address(0), address(
82             this), _addrOwner);
83         IIndex(oldIndexOwner).destruct();
84         address oldIndexOwnerRoot = resolveIndex(_addrRoot, address
85             (this), _addrOwner);
86         IIndex(oldIndexOwnerRoot).destruct();
87
88         recipient.transfer(0, false, 64);
89         selfdestruct(recipient);
90     }

```

7.6.2 Function getInfo

- TODO

```

94     function getInfo() public view override
95     returns(
96         mapping(uint128 => bytes) content,
97         address author,
98         string name,
99         string description,
100         string tokenCode,
101         uint64 creationDate,
102         string comment
103     ) {
104         content = _content;

```

```

105     author = _addrAuthor;
106     name = _name;
107     description = _description;
108     tokenCode = _tokenCode;
109     creationDate = _creationDate;
110     comment = _comment;
111 }

```

7.6.3 Function getOwner

- TODO

```

89     function getOwner() public view override returns(address
90         addrOwner, address addrNftData) {
91         addrOwner = _addrOwner;
92         addrNftData = address(this);
93     }

```

7.7 Internal Method Definitions

7.7.1 Function deployIndex

- TODO

```

63     function deployIndex(address owner) private {
64         TvmCell codeIndexOwner = _buildIndexCode(address(0), owner)
65         ;
66         TvmCell stateIndexOwner = _buildIndexState(codeIndexOwner,
67             address(this));
68         new Index
69         {stateInit: stateIndexOwner, value: Constants.
70             DEPLOY_INDEX_FEE, flag: 0}
71         (_addrRoot);
72
73         TvmCell codeIndexOwnerRoot = _buildIndexCode(_addrRoot,
74             owner);
75         TvmCell stateIndexOwnerRoot = _buildIndexState(
76             codeIndexOwnerRoot, address(this));
77         new Index
78         {stateInit: stateIndexOwnerRoot, value: Constants.
79             DEPLOY_INDEX_FEE, flag: 0}
80         (_addrRoot);
81     }

```


Chapter 8

Contract DataResolver

Contents

8.1	Overview	32
8.2	Variable Definitions	32
8.3	Public Method Definitions	33
8.3.1	Function resolveCodeHashData	33
8.3.2	Function resolveData	33
8.4	Internal Method Definitions	33
8.4.1	Function _buildDataCode	33
8.4.2	Function _buildDataState	33

8.1 Overview

In file `DataResolver.sol`

8.2 Variable Definitions

TvmCell	_codeData	
		assigned in @2.NftRoot.:constructor
		used in @2.NftRoot.:constructor
		used in @7.DataResolver._buildDataCode

11 TvmCell _codeData;

8.3 Public Method Definitions

8.3.1 Function resolveCodeHashData

- TODO

```

13     function resolveCodeHashData() public view returns (uint256
14         codeHashData) {
15         return tvn.hash(_buildDataCode(address(this)));
16     }

```

8.3.2 Function resolveData

- TODO

```

17     function resolveData(
18         address addrRoot,
19         uint256 id
20     ) public view returns (address addrData) {
21         TvmCell code = _buildDataCode(addrRoot);
22         TvmCell state = _buildDataState(code, id);
23         uint256 hashState = tvn.hash(state);
24         addrData = address.makeAddrStd(0, hashState);
25     }

```

8.4 Internal Method Definitions

8.4.1 Function _buildDataCode

- TODO

```

27     function _buildDataCode(address addrRoot) internal virtual view
28         returns (TvmCell) {
29         TvmBuilder salt;
30         salt.store(addrRoot);
31         return tvn.setCodeSalt(_codeData, salt.toCell());
32     }

```

8.4.2 Function _buildDataState

- TODO

```

33     function _buildDataState(
34         TvmCell code,
35         uint256 id
36     ) internal virtual pure returns (TvmCell) {
37         return tvn.buildStateInit({
38             contr: Data,

```

```
39         varInit: {_id: id},  
40         code: code  
41     });  
42 }
```

Chapter 9

Contract Index

Contents

9.1	Overview	35
9.2	Contract Inheritance	35
9.3	Static Variable Definitions	35
9.4	Variable Definitions	36
9.5	Constructor Definitions	36
9.5.1	Constructor	36
9.6	Public Method Definitions	37
9.6.1	Function destruct	37
9.6.2	Function getInfo	37

9.1 Overview

In file `Index.sol`

9.2 Contract Inheritance

IIndex	
--------	--

9.3 Static Variable Definitions

address	_addrData	
		used in @10.Index.getInfo
		used in @10.Index.destruct
		used in @10.Index.destruct
		used in @10.Index.constructor

```
13     address static _addrData;
```

9.4 Variable Definitions

address	_addrRoot	
		used in @10.Index.getInfo
		assigned in @10.Index.:constructor
		used in @10.Index.:constructor
		assigned in @10.Index.:constructor
		used in @10.Index.:constructor
address	_addrOwner	
		used in @10.Index.getInfo
		assigned in @10.Index.:constructor
		used in @10.Index.:constructor

```
11     address _addrRoot;
```

```
12     address _addrOwner;
```

9.5 Constructor Definitions

9.5.1 Constructor

Critical issue: Constructor for Index (fake)

lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
 ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
 lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
 lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum
 ipsum lorem ipsum lorem ipsum

- TODO

```
15     constructor(address root) public {
16         optional(TvmCell) optSalt = tvn.codeSalt(tvn.code());
17         require(optSalt.hasValue(), Errors.ERROR_EMPTY_SALT);
18         (address addrRoot, address addrOwner) = optSalt
19             .get()
20             .toSlice()
21             .decode(address, address);
22         require(msg.sender == _addrData, Errors.
23             ERROR_MESSAGE_SENDER_IS_NOT_OWNER);
23         tvn.accept();
24         _addrRoot = addrRoot;
```

```
25     _addrOwner = addrOwner;
26     if(addrRoot == address(0)) {
27         _addrRoot = root;
28     }
29 }
```

9.6 Public Method Definitions

9.6.1 Function destruct

- TODO

```
41     function destruct() public override {
42         require(msg.sender == _addrData, Errors.
43             ERROR_MESSAGE_SENDER_IS_NOT_OWNER);
44         selfdestruct(_addrData);
45     }
```

9.6.2 Function getInfo

- TODO

```
31     function getInfo() public view override returns (
32         address addrRoot,
33         address addrOwner,
34         address addrData
35     ) {
36         addrRoot = _addrRoot;
37         addrOwner = _addrOwner;
38         addrData = _addrData;
39     }
```

Chapter 10

Contract IndexBasis

Contents

10.1 Overview	38
10.2 Static Variable Definitions	38
10.3 Modifier Definitions	39
10.3.1 Modifier onlyRoot	39
10.4 Constructor Definitions	39
10.4.1 Constructor	39
10.5 Public Method Definitions	39
10.5.1 Function destruct	39
10.5.2 Function getInfo	39

10.1 Overview

In file `IndexBasis.sol`

10.2 Static Variable Definitions

address	_addrRoot	
		used in @5.IndexBasis.getInfo
		used in @5.IndexBasis.destruct
uint256	_codeHashData	
		used in @5.IndexBasis.getInfo

```
9     address static _addrRoot;
```

```
10    uint256 static _codeHashData;
```

10.3 Modifier Definitions

10.3.1 Modifier onlyRoot

```

12     modifier onlyRoot() {
13         require(msg.sender == _addrRoot, Errors.
14             ERROR_MESSAGE_SENDER_IS_NOT_ROOT);
15         tvmm.accept();
16     }

```

10.4 Constructor Definitions

10.4.1 Constructor

Critical issue: Constructor for IndexBasis (fake)

loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum
 ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum
 loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum
 loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren
 ipsum loren ipsum loren ipsum

- TODO

```

18     constructor() public onlyRoot {}

```

10.5 Public Method Definitions

10.5.1 Function destruct

- TODO

```

25     function destruct() public onlyRoot {
26         selfdestruct(_addrRoot);
27     }

```

10.5.2 Function getInfo

- TODO

```

20     function getInfo() public view returns (address addrRoot,
21         uint256 codeHashData) {
22         addrRoot = _addrRoot;
23         codeHashData = _codeHashData;
24     }

```


Chapter 11

Contract IndexResolver

Contents

11.1 Overview	40
11.2 Variable Definitions	40
11.3 Public Method Definitions	41
11.3.1 Function resolveCodeHashIndex	41
11.3.2 Function resolveIndex	41
11.4 Internal Method Definitions	41
11.4.1 Function _buildIndexCode	41
11.4.2 Function _buildIndexState	41

11.1 Overview

In file `IndexResolver.sol`

11.2 Variable Definitions

TvmCell	_codeIndex	
		used in @2.NftRoot.mintNft
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor
		assigned in @9.Data.:constructor
		used in @9.Data.:constructor
		used in @8.IndexResolver._buildIndexCode

11 TvmCell _codeIndex;

11.3 Public Method Definitions

11.3.1 Function resolveCodeHashIndex

- TODO

```

13     function resolveCodeHashIndex(
14         address addrRoot,
15         address addrOwner
16     ) public view returns (uint256 codeHashIndex) {
17         return tvn.hash(_buildIndexCode(addrRoot, addrOwner));
18     }

```

11.3.2 Function resolveIndex

- TODO

```

20     function resolveIndex(
21         address addrRoot,
22         address addrData,
23         address addrOwner
24     ) public view returns (address addrIndex) {
25         TvmCell code = _buildIndexCode(addrRoot, addrOwner);
26         TvmCell state = _buildIndexState(code, addrData);
27         uint256 hashState = tvn.hash(state);
28         addrIndex = address.makeAddrStd(0, hashState);
29     }

```

11.4 Internal Method Definitions

11.4.1 Function _buildIndexCode

- TODO

```

31     function _buildIndexCode(
32         address addrRoot,
33         address addrOwner
34     ) internal virtual view returns (TvmCell) {
35         TvmBuilder salt;
36         salt.store(addrRoot);
37         salt.store(addrOwner);
38         return tvn.setCodeSalt(_codeIndex, salt.toCell());
39     }

```

11.4.2 Function _buildIndexState

- TODO

```
41     function _buildIndexState(  
42         TvmCell code,  
43         address addrData  
44     ) internal virtual pure returns (TvmCell) {  
45         return tvm.buildStateInit({  
46             contr: Index,  
47             varInit: {_addrData: addrData},  
48             code: code  
49         });  
50     }
```

Chapter 12

Contract Manager

Contents

12.1 Overview	43
12.2 Variable Definitions	43
12.3 Constructor Definitions	44
12.3.1 Constructor	44
12.4 Public Method Definitions	44
12.4.1 Function <code>deployRoot</code>	44
12.5 Internal Method Definitions	44
12.5.1 Function <code>_buildNftRootState</code>	44

12.1 Overview

In file `Manager.sol`

12.2 Variable Definitions

<code>TvmCell</code>	<code>_rootCode</code>	
		used in <code>@1.Manager._buildNftRootState</code>
		assigned in <code>@1.Manager.constructor</code>
		used in <code>@1.Manager.constructor</code>

13 `TvmCell _rootCode;`

12.3.1 Constructor

12.3.1 Constructor

[illegible]

- ```
15 constructor (TvmCell rootCode) public {
16 tvml.accept();
17 _rootCode = rootCode;
18 }
```

### 12.4.1 Function deployRoot

- ```

20     function deployRoot(
21         address addrOwner,
22         TvmCell codeIndex,
23         TvmCell codeData,
24         string name,
25         string description,
26         string tokenCode,
27         uint256 totalSupply,
28         uint128 index,
29         bytes part
30     ) public view {
31         tvn.accept();
32
33         TvmCell stateNftRoot = _buildNftRootState(addrOwner);
34         new NftRoot {stateInit: stateNftRoot, value: Constants.
            DEPLOY_INDEX_FEE}( codeIndex, codeData, name,
            description, tokenCode, totalSupply, index, part);
35     }

```

12.5.1 Function `_buildNftRootState`

- 44

```
37     function _buildNftRootState( address addrOwner) internal
38         virtual view returns (TvmCell) {
39             TvmCell code = _rootCode.toSlice().loadRef();
40             return tvm.buildStateInit({
41                 contr: NftRoot,
42                 varInit: {_addrOwner: addrOwner},
43                 code: code
44             });
45     }
```

Chapter 13

Contract NftRoot

Contents

13.1 Overview	46
13.2 Contract Inheritance	46
13.3 Static Variable Definitions	47
13.4 Variable Definitions	49
13.5 Modifier Definitions	50
13.5.1 Modifier onlyOwner	50
13.6 Constructor Definitions	50
13.6.1 Constructor	50
13.7 Public Method Definitions	51
13.7.1 Function burn	51
13.7.2 Function deployBasis	51
13.7.3 Function destructBasis	52
13.7.4 Function getInfo	52
13.7.5 Function mintNft	52
13.7.6 Function setPrice	53

13.1 Overview

In file `NftRoot.sol`

13.2 Contract Inheritance

DataResolver	
IndexResolver	

13.3 Static Variable Definitions

address	_addrOwner	
---------	------------	--

32 `address static _addrOwner;`

13.4 Variable Definitions

uint256	_totalMinted	
		assigned in @2.NftRoot.mintNft
		used in @2.NftRoot.mintNft
		used in @2.NftRoot.mintNft
		used in @2.NftRoot.mintNft
address	_addrBasis	
		used in @2.Nft-Root.destructBasis
		assigned in @2.Nft-Root.deployBasis
		used in @2.NftRoot.deployBasis
uint256	_totalSupply	
		used in @2.NftRoot.mintNft
		used in @2.NftRoot.getInfo
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor
string	_name	
		used in @2.NftRoot.mintNft
		used in @2.NftRoot.getInfo
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor
string	_description	
		used in @2.NftRoot.mintNft
		used in @2.NftRoot.getInfo
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor
string	_tokenCode	
		used in @2.NftRoot.mintNft
		used in @2.NftRoot.getInfo
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor
mapping (uint128 => bytes)	_content	
		used in @2.NftRoot.mintNft
		used in @2.NftRoot.getInfo
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor
uint128	_price	
		assigned in @2.NftRoot.setPrice
		used in @2.NftRoot.setPrice
CHAPTER 13. CONTRACT NFTROOT		used in @2.NftRoot.getInfo 49
		used in @2.NftRoot.burn
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor

```
18     uint256 _totalMinted;
19     address _addrBasis;
21     uint256 _totalSupply;
23     string _name;
24     string _description;
25     string _tokenCode;
28     mapping(uint128 => bytes) _content;
30     uint128 _price;
```

13.5 Modifier Definitions

13.5.1 Modifier onlyOwner

```

34     modifier onlyOwner() {
35         require(msg.sender == _addrOwner, Errors.
36             ERROR_MESSAGE_SENDER_IS_NOT_OWNER);
37         tvmm.accept();
38     }

```

13.6 Constructor Definitions

13.6.1 Constructor

Critical issue: Constructor for NftRoot (fake)

loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren
ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum
loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum
loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren ipsum loren
ipsum loren ipsum loren ipsum

- TODO

```

40     constructor(
41         TvmCell codeIndex,
42         TvmCell codeData,
43         string name,
44         string description,
45         string tokenCode,
46         uint256 totalSupply,
47         uint128 index,
48         bytes part
49     ) public {

```

```

50     tvm.accept();
51     _codeIndex = codeIndex;
52     _codeData = codeData;
53     _name = name;
54     _description = description;
55     _tokenCode = tokenCode;
56     _totalSupply = totalSupply;
57
58     _content[index] = part;
59
60     _price = 1 ton;
61 }

```

13.7 Public Method Definitions

13.7.1 Function burn

- TODO

```

123     function burn(address dataAddress, address owner) public
124         onlyOwner {
125             require(msg.value >= (_price), Errors.
126                 ERROR_MSG_VALUE_LESS_THAN_PRICE);
127
128             Data(dataAddress).destruct
129                 {value: msg.value, flag: 3, bounce: true}
130                 (owner);
131 }

```

13.7.2 Function deployBasis

- TODO

```

85     function deployBasis(TvmCell codeIndexBasis) public onlyOwner {
86         require(msg.value > 0.5 ton, Errors.ERROR_NOT_ENOUGH_GRAMS)
87         ;
88         uint256 codeHasData = resolveCodeHashData();
89         TvmCell state = tvm.buildStateInit({
90             contr: IndexBasis,
91             varInit: {
92                 _codeHashData: codeHasData,
93                 _addrRoot: address(this)
94             },
95             code: codeIndexBasis
96         });
97         _addrBasis = new IndexBasis{stateInit: state, value: 0.4
98             ton}();
99 }

```

13.7.3 Function destructBasis

- TODO

```

99     function destructBasis() public view onlyOwner {
100         IIndexBasis(_addrBasis).destruct();
101     }

```

13.7.4 Function getInfo

- TODO

```

103     function getInfo() public view returns (
104         mapping(uint128 => bytes) content,
105         string name,
106         string description,
107         string tokenCode,
108         uint256 totalSupply,
109         uint128 price
110     ) {
111         content = _content;
112         name = _name;
113         description = _description;
114         tokenCode = _tokenCode;
115         totalSupply = _totalSupply;
116         price = _price;
117     }

```

13.7.5 Function mintNft

- TODO

```

63     function mintNft(uint64 creationDate, string comment, address
64         owner) public onlyOwner {
65         require(msg.value >= 1.6 ton, Errors.ERROR_NOT_ENOUGH_GRAMS);
66         require(_totalMinted <= _totalSupply, Errors.ERROR_MINTED_TOO_MUCH);
67         TvmCell codeData = _buildDataCode(address(this));
68         TvmCell stateData = _buildDataState(codeData, _totalMinted);
69         new Data
70             {stateInit: stateData, value: 1.5 ton} (
71             owner,
72             _codeIndex,
73             msg.sender,
74             _name,
75             _description,
76             _tokenCode,
77             creationDate,
78             comment,
79             0,
80             _content[0]

```

```
80         );  
81  
82         _totalMinted++;  
83     }
```

13.7.6 Function setPrice

- TODO

```
119     function setPrice(uint128 price) public onlyOwner {  
120         _price = price;  
121     }
```