

Audit

By OCamlPro

• September 7, 2021

# Table of Major and Critical Issues

Critical issue: Constructor for Data (fake) . . . . .	28
Critical issue: Constructor for Index (fake) . . . . .	35
Critical issue: Constructor for IndexBasis (fake) . . . . .	38
Critical issue: Constructor for Manager (fake) . . . . .	43
Critical issue: Constructor for NftRoot (fake) . . . . .	49

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Overview</b>	<b>8</b>
<b>3</b>	<b>Library Modules</b>	<b>9</b>
3.1	Module "Constants.sol" . . . . .	10
3.1.1	Pragmas . . . . .	10
3.1.2	Contract Definitions . . . . .	10
3.2	Module "Errors.sol" . . . . .	11
3.2.1	Pragmas . . . . .	11
3.2.2	Contract Definitions . . . . .	11
3.3	Module "true_nft_audit.sol" . . . . .	12
3.3.1	Imports . . . . .	12
<b>4</b>	<b>Interface Modules</b>	<b>13</b>
4.1	Module "IData.sol" . . . . .	14
4.1.1	Pragmas . . . . .	14
4.1.2	Contract Definitions . . . . .	14
4.2	Module "IIndex.sol" . . . . .	15
4.2.1	Pragmas . . . . .	15
4.2.2	Contract Definitions . . . . .	15
4.3	Module "IIndexBasis.sol" . . . . .	16
4.3.1	Pragmas . . . . .	16
4.3.2	Contract Definitions . . . . .	16
<b>5</b>	<b>Contract Modules</b>	<b>17</b>
5.1	Module "Data.sol" . . . . .	18
5.1.1	Pragmas . . . . .	18
5.1.2	Imports . . . . .	18
5.1.3	Contract Definitions . . . . .	18
5.2	Module "DataResolver.sol" . . . . .	19
5.2.1	Pragmas . . . . .	19
5.2.2	Imports . . . . .	19
5.2.3	Contract Definitions . . . . .	19

5.3	Module "Index.sol" . . . . .	20
5.3.1	Pragmas . . . . .	20
5.3.2	Imports . . . . .	20
5.3.3	Contract Definitions . . . . .	20
5.4	Module "IndexBasis.sol" . . . . .	21
5.4.1	Pragmas . . . . .	21
5.4.2	Imports . . . . .	21
5.4.3	Contract Definitions . . . . .	21
5.5	Module "IndexResolver.sol" . . . . .	22
5.5.1	Pragmas . . . . .	22
5.5.2	Imports . . . . .	22
5.5.3	Contract Definitions . . . . .	22
5.6	Module "Manager.sol" . . . . .	23
5.6.1	Pragmas . . . . .	23
5.6.2	Imports . . . . .	23
5.6.3	Contract Definitions . . . . .	23
5.7	Module "NftRoot.sol" . . . . .	24
5.7.1	Pragmas . . . . .	24
5.7.2	Imports . . . . .	24
5.7.3	Contract Definitions . . . . .	24
<b>6</b>	<b>Contract Data</b>	<b>25</b>
6.1	Overview . . . . .	25
6.2	Contract Inheritance . . . . .	25
6.3	Static Variable Definitions . . . . .	25
6.4	Variable Definitions . . . . .	27
6.5	Constructor Definitions . . . . .	28
6.5.1	Constructor . . . . .	28
6.6	Public Method Definitions . . . . .	29
6.6.1	Function destruct . . . . .	29
6.6.2	Function getInfo . . . . .	29
6.6.3	Function getOwner . . . . .	30
6.7	Internal Method Definitions . . . . .	30
6.7.1	Function deployIndex . . . . .	30
<b>7</b>	<b>Contract DataResolver</b>	<b>31</b>
7.1	Overview . . . . .	31
7.2	Variable Definitions . . . . .	31
7.3	Public Method Definitions . . . . .	32
7.3.1	Function resolveCodeHashData . . . . .	32
7.3.2	Function resolveData . . . . .	32
7.4	Internal Method Definitions . . . . .	32
7.4.1	Function _buildDataCode . . . . .	32
7.4.2	Function _buildDataState . . . . .	32

<b>8</b>	<b>Contract Index</b>	<b>34</b>
8.1	Overview . . . . .	34
8.2	Contract Inheritance . . . . .	34
8.3	Static Variable Definitions . . . . .	34
8.4	Variable Definitions . . . . .	35
8.5	Constructor Definitions . . . . .	35
8.5.1	Constructor . . . . .	35
8.6	Public Method Definitions . . . . .	36
8.6.1	Function destruct . . . . .	36
8.6.2	Function getInfo . . . . .	36
<b>9</b>	<b>Contract IndexBasis</b>	<b>37</b>
9.1	Overview . . . . .	37
9.2	Static Variable Definitions . . . . .	37
9.3	Modifier Definitions . . . . .	38
9.3.1	Modifier onlyRoot . . . . .	38
9.4	Constructor Definitions . . . . .	38
9.4.1	Constructor . . . . .	38
9.5	Public Method Definitions . . . . .	38
9.5.1	Function destruct . . . . .	38
9.5.2	Function getInfo . . . . .	38
<b>10</b>	<b>Contract IndexResolver</b>	<b>39</b>
10.1	Overview . . . . .	39
10.2	Variable Definitions . . . . .	39
10.3	Public Method Definitions . . . . .	40
10.3.1	Function resolveCodeHashIndex . . . . .	40
10.3.2	Function resolveIndex . . . . .	40
10.4	Internal Method Definitions . . . . .	40
10.4.1	Function _buildIndexCode . . . . .	40
10.4.2	Function _buildIndexState . . . . .	40
<b>11</b>	<b>Contract Manager</b>	<b>42</b>
11.1	Overview . . . . .	42
11.2	Variable Definitions . . . . .	42
11.3	Constructor Definitions . . . . .	43
11.3.1	Constructor . . . . .	43
11.4	Public Method Definitions . . . . .	43
11.4.1	Function deployRoot . . . . .	43
11.5	Internal Method Definitions . . . . .	43
11.5.1	Function _buildNftRootState . . . . .	43

<b>12 Contract NftRoot</b>	<b>45</b>
12.1 Overview . . . . .	45
12.2 Contract Inheritance . . . . .	45
12.3 Static Variable Definitions . . . . .	46
12.4 Variable Definitions . . . . .	48
12.5 Modifier Definitions . . . . .	49
12.5.1 Modifier onlyOwner . . . . .	49
12.6 Constructor Definitions . . . . .	49
12.6.1 Constructor . . . . .	49
12.7 Public Method Definitions . . . . .	50
12.7.1 Function burn . . . . .	50
12.7.2 Function deployBasis . . . . .	50
12.7.3 Function destructBasis . . . . .	51
12.7.4 Function getInfo . . . . .	51
12.7.5 Function mintNft . . . . .	51
12.7.6 Function setPrice . . . . .	52

# To edit this document

In the `report.tex` file, choose:

- `\soldraftfalse` to remove draft mode (watermarks, advises)
- `\solmoduletrue` to display modules by chapter instead of contracts
- `\soltabletrue` 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}`

## Chapter 1

# Introduction



## Chapter 2

### Overview

## Chapter 3

# Library Modules

## 3.1 Module "Constants.sol"

### 3.1.1 Pragas

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

### 3.1.2 Contract Definitions

- Constants

## 3.2 Module "Errors.sol"

### 3.2.1 Pragas

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

### 3.2.2 Contract Definitions

- Errors

## 3.3 Module "true\_nft\_audit.sol"

### 3.3.1 Imports

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

## Chapter 4

# Interface Modules

## 4.1 Module "IData.sol"

### 4.1.1 Pragas

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

### 4.1.2 Contract Definitions

- IData

## 4.2 Module "IIndex.sol"

### 4.2.1 Pragas

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

### 4.2.2 Contract Definitions

- IIndex



## 4.3 Module "IIndexBasis.sol"

### 4.3.1 Pragas

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

### 4.3.2 Contract Definitions

- IIndexBasis

## Chapter 5

# Contract Modules

## 5.1 Module "Data.sol"

### 5.1.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

### 5.1.2 Imports

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

### 5.1.3 Contract Definitions

- Data

## 5.2 Module "DataResolver.sol"

### 5.2.1 Pragas

ton	-solidity >= 0.43.0	
AbiHeader	expire	
AbiHeader	time	

### 5.2.2 Imports

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

### 5.2.3 Contract Definitions

- DataResolver

## 5.3 Module "Index.sol"

### 5.3.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

### 5.3.2 Imports

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

### 5.3.3 Contract Definitions

- Index

## 5.4 Module "IndexBasis.sol"

### 5.4.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

### 5.4.2 Imports

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

### 5.4.3 Contract Definitions

- IndexBasis

## 5.5 Module "IndexResolver.sol"

### 5.5.1 Pragas

ton	-solidity >= 0.43.0	
AbiHeader	expire	
AbiHeader	time	

### 5.5.2 Imports

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

### 5.5.3 Contract Definitions

- IndexResolver

## 5.6 Module "Manager.sol"

### 5.6.1 Pragas

ton	-solidity >= 0.43.0	
AbiHeader	expire	
AbiHeader	time	

### 5.6.2 Imports

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

### 5.6.3 Contract Definitions

- Manager



## 5.7 Module "NftRoot.sol"

### 5.7.1 Pragas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

### 5.7.2 Imports

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

### 5.7.3 Contract Definitions

- NftRoot

# Chapter 6

## Contract Data

### Contents

<b>6.1</b>	<b>Overview</b>	<b>25</b>
<b>6.2</b>	<b>Contract Inheritance</b>	<b>25</b>
<b>6.3</b>	<b>Static Variable Definitions</b>	<b>25</b>
<b>6.4</b>	<b>Variable Definitions</b>	<b>27</b>
<b>6.5</b>	<b>Constructor Definitions</b>	<b>28</b>
6.5.1	Constructor	28
<b>6.6</b>	<b>Public Method Definitions</b>	<b>29</b>
6.6.1	Function destruct	29
6.6.2	Function getInfo	29
6.6.3	Function getOwner	30
<b>6.7</b>	<b>Internal Method Definitions</b>	<b>30</b>
6.7.1	Function deployIndex	30

### 6.1 Overview

In file `Data.sol`

### 6.2 Contract Inheritance

IData	
IndexResolver	

### 6.3 Static Variable Definitions

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

```
17  uint256 static _id;
```

## 6.4 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;
```

## 6.5 Constructor Definitions

### 6.5.1 Constructor

**Critical issue: Constructor for Data (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

```

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 = tvn.codeSalt(tvn.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 }

```

## 6.6 Public Method Definitions

### 6.6.1 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     }

```

### 6.6.2 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 }

```

### 6.6.3 Function getOwner

- TODO

```

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

```

## 6.7 Internal Method Definitions

### 6.7.1 Function deployIndex

- TODO

```

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

```

## Chapter 7

# Contract DataResolver

### Contents

<b>7.1 Overview</b>	<b>31</b>
<b>7.2 Variable Definitions</b>	<b>31</b>
<b>7.3 Public Method Definitions</b>	<b>32</b>
7.3.1 Function resolveCodeHashData	32
7.3.2 Function resolveData	32
<b>7.4 Internal Method Definitions</b>	<b>32</b>
7.4.1 Function _buildDataCode	32
7.4.2 Function _buildDataState	32

### 7.1 Overview

In file `DataResolver.sol`

### 7.2 Variable Definitions

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

11 TvmCell \_codeData;



## 7.3 Public Method Definitions

### 7.3.1 Function resolveCodeHashData

- TODO

```

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

```

### 7.3.2 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     }

```

## 7.4 Internal Method Definitions

### 7.4.1 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     }

```

### 7.4.2 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 8

# Contract Index

### Contents

<b>8.1</b>	<b>Overview</b>	<b>34</b>
<b>8.2</b>	<b>Contract Inheritance</b>	<b>34</b>
<b>8.3</b>	<b>Static Variable Definitions</b>	<b>34</b>
<b>8.4</b>	<b>Variable Definitions</b>	<b>35</b>
<b>8.5</b>	<b>Constructor Definitions</b>	<b>35</b>
8.5.1	Constructor	35
<b>8.6</b>	<b>Public Method Definitions</b>	<b>36</b>
8.6.1	Function destruct	36
8.6.2	Function getInfo	36

## 8.1 Overview

In file `Index.sol`

## 8.2 Contract Inheritance

IIndex	
--------	--

## 8.3 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;
```

## 8.4 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;
```

## 8.5 Constructor Definitions

### 8.5.1 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 = tvm.codeSalt(tvm.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         tvm.accept();
24         _addrRoot = addrRoot;
```

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

## 8.6 Public Method Definitions

### 8.6.1 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     }
```

### 8.6.2 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 9

# Contract IndexBasis

### Contents

---

<b>9.1 Overview</b>	<b>37</b>
<b>9.2 Static Variable Definitions</b>	<b>37</b>
<b>9.3 Modifier Definitions</b>	<b>38</b>
9.3.1 Modifier onlyRoot	38
<b>9.4 Constructor Definitions</b>	<b>38</b>
9.4.1 Constructor	38
<b>9.5 Public Method Definitions</b>	<b>38</b>
9.5.1 Function destruct	38
9.5.2 Function getInfo	38

---

### 9.1 Overview

In file `IndexBasis.sol`

### 9.2 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;
```

### 9.3.1 Modifier onlyRoot

## 9.4 Constructor Definitions

**Critical issue: Constructor for IndexBasis (fake)**

- TODO

## 9.5 Public Method Definitions

- TODO

### 9.5.2 Function getInfo

- TODO

CHAPTER 9. CONTRACT INDEXBASIS

# Chapter 10

## Contract IndexResolver

### Contents

<b>10.1 Overview</b>	<b>39</b>
<b>10.2 Variable Definitions</b>	<b>39</b>
<b>10.3 Public Method Definitions</b>	<b>40</b>
10.3.1 Function resolveCodeHashIndex	40
10.3.2 Function resolveIndex	40
<b>10.4 Internal Method Definitions</b>	<b>40</b>
10.4.1 Function _buildIndexCode	40
10.4.2 Function _buildIndexState	40

### 10.1 Overview

In file `IndexResolver.sol`

### 10.2 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;



## 10.3 Public Method Definitions

### 10.3.1 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     }

```

### 10.3.2 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     }

```

## 10.4 Internal Method Definitions

### 10.4.1 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     }

```

### 10.4.2 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 11

## Contract Manager

### Contents

---

<b>11.1 Overview</b>	<b>42</b>
<b>11.2 Variable Definitions</b>	<b>42</b>
<b>11.3 Constructor Definitions</b>	<b>43</b>
11.3.1 Constructor	43
<b>11.4 Public Method Definitions</b>	<b>43</b>
11.4.1 Function deployRoot	43
<b>11.5 Internal Method Definitions</b>	<b>43</b>
11.5.1 Function _buildNftRootState	43

---

### 11.1 Overview

In file `Manager.sol`

### 11.2 Variable Definitions

TvmCell	_rootCode	
		used in @1.Manager._buildNftRootState
		assigned in @1.Manager.constructor
		used in @1.Manager.constructor

13 `TvmCell _rootCode;`

### 11.3.1 Constructor

### 11.3.1 Constructor

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  
ipsum lorem ipsum lorem ipsum

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

### 11.4.1 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     }

```

### 11.5.1 Function `_buildNftRootState`

- ## CHAPTER 11. CONTRACT MANAGER

```
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 12

# Contract NftRoot

### Contents

---

<b>12.1 Overview</b>	<b>45</b>
<b>12.2 Contract Inheritance</b>	<b>45</b>
<b>12.3 Static Variable Definitions</b>	<b>46</b>
<b>12.4 Variable Definitions</b>	<b>48</b>
<b>12.5 Modifier Definitions</b>	<b>49</b>
12.5.1 Modifier onlyOwner	49
<b>12.6 Constructor Definitions</b>	<b>49</b>
12.6.1 Constructor	49
<b>12.7 Public Method Definitions</b>	<b>50</b>
12.7.1 Function burn	50
12.7.2 Function deployBasis	50
12.7.3 Function destructBasis	51
12.7.4 Function getInfo	51
12.7.5 Function mintNft	51
12.7.6 Function setPrice	52

---

## 12.1 Overview

In file `NftRoot.sol`

## 12.2 Contract Inheritance

DataResolver	
IndexResolver	

## 12.3 Static Variable Definitions

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

32     `address static _addrOwner;`





## 12.4 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 12. CONTRACT NFTROOT		used in @2.NftRoot.getInfo 48
		used in @2.NftRoot.burn
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor



```

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 }

```

## 12.7 Public Method Definitions

### 12.7.1 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 }

```

### 12.7.2 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 }

```

### 12.7.3 Function destructBasis

- TODO

```

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

```

### 12.7.4 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     }

```

### 12.7.5 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     }
```

### 12.7.6 Function setPrice

- TODO

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