Audit

By OCamlPro

September 7, 2021

Table of Major and Critical Issues

Contents

To edit this document

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}**

Chapter 1

Introduction

Chapter 2

Overview

Chapter 3

Library Modules

3.1 Module "Constants.sol"

3.1.1 Pragmas

ton	-solidity >= 0.43.0	

3.1.2 Contract Definitions

- Constants

3.2 Module "Errors.sol"

3.2.1 Pragmas

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 Pragmas

ton	-solidity >= 0.43.0	

4.1.2 Contract Definitions

- IData

4.2 Module ”IIndex.sol”

4.2.1 Pragmas

ton	-solidity >= 0.43.0	

4.2.2 Contract Definitions

- IIndex

4.3 Module ”IIndexBasis.sol”

4.3.1 Pragmas

ton	-solidity >= 0.43.0	

4.3.2 Contract Definitions

- IIndexBasis

Chapter 5

Contract Modules

5.1 Module "Data.sol"

5.1.1 Pragmas

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 Pragmas

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 Pragmas

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 Pragmas

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 Pragmas

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 Pragmas

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 Pragmas

ton	-solidity >=0.43.0	
AbiHeader	expire	
AbiHeader	time	

5.7.2 Imports

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

5.7.3 Contract Definitions

- NftRoot

Chapter 6

Contract Data

Contents

6.1 Overview

In file `Data.sol`

6.2 Contract Inheritance

IData	
IndexResolver	

6.3 Static Variable Definitions

uint256	_id	

```
    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

```
    address _addrRoot;
```

```
    address _addrOwner;
```

```
    address _addrAuthor;
```

```
    string _name;
```

```
    string _description;
```

```
    string _tokenCode;
```

```
    uint64 _creationDate;
```

```
    string _comment;
```

```
    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

```
    constructor(
        address addrOwner,
        TvmCell codeIndex,
        address addrAuthor,
        string name,
        string description,
        string tokenCode,
        uint64 creationDate,
        string comment,
        uint128 index,
        bytes part
    ) public {
        optional(TvmCell) optSalt = tvm.codeSalt(tvm.code());
        require(optSalt.hasValue(), Errors.ERROR_EMPTY_SALT);
        (address addrRoot) = optSalt
            .get()
            .toSlice()
            .decode(address);
        require(msg.sender == addrRoot, Errors.
            ERROR_MESSAGE_SENDER_IS_NOT_ROOT);
        require(msg.value >= Constants.MIN_FOR_DEPLOY);
        _addrRoot = addrRoot;
        _addrOwner = addrOwner;
        _addrAuthor = addrAuthor;
        _name = name;
        _description = description;
        _tokenCode = tokenCode;
        _creationDate = creationDate;
        _comment = comment;
        _codeIndex = codeIndex;

        _content[index] = part;

        deployIndex(addrOwner);
    }
```

6.6 Public Method Definitions

6.6.1 Function destruct

- TODO

```
    function destruct(address recipient) public {
        require(msg.sender == _addrRoot, Errors.
            ERROR_MESSAGE_SENDER_IS_NOT_ROOT);

        address oldIndexOwner = resolveIndex(address(0), address(
            this), _addrOwner);
        IIndex(oldIndexOwner).destruct();
        address oldIndexOwnerRoot = resolveIndex(_addrRoot, address
            (this), _addrOwner);
        IIndex(oldIndexOwnerRoot).destruct();

        recipient.transfer(0, false, 64);
        selfdestruct(recipient);
    }
```

6.6.2 Function getInfo

- TODO

```
    function getInfo() public view override
    returns(
        mapping(uint128 => bytes) content,
        address author,
        string name,
        string description,
        string tokenCode,
        uint64 creationDate,
        string comment
    ) {
        content = _content;
        author = _addrAuthor;
        name = _name;
        description = _description;
        tokenCode = _tokenCode;
        creationDate = _creationDate;
        comment = _comment;
    }
```

6.6.3 Function getOwner

- TODO

```
    function getOwner() public view override returns(address
        addrOwner, address addrNftData) {
        addrOwner = _addrOwner;
        addrNftData = address(this);
    }
```

6.7 Internal Method Definitions

6.7.1 Function deployIndex

- TODO

```
    function deployIndex(address owner) private {
        TvmCell codeIndexOwner = _buildIndexCode(address(0), owner)
            ;
        TvmCell stateIndexOwner = _buildIndexState(codeIndexOwner,
            address(this));
        new Index
            {stateInit: stateIndexOwner, value: Constants.
                DEPLOY_INDEX_FEE, flag: 0}
            (_addrRoot);

        TvmCell codeIndexOwnerRoot = _buildIndexCode(_addrRoot,
            owner);
        TvmCell stateIndexOwnerRoot = _buildIndexState(
            codeIndexOwnerRoot, address(this));
        new Index
            {stateInit: stateIndexOwnerRoot, value: Constants.
                DEPLOY_INDEX_FEE, flag: 0}
            (_addrRoot);
    }
```

Chapter 7

Contract DataResolver

Contents

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

```
    TvmCell _codeData;
```

7.3 Public Method Definitions

7.3.1 Function resolveCodeHashData

- TODO

```
    function resolveCodeHashData() public view returns (uint256
        codeHashData) {
        return tvm.hash(_buildDataCode(address(this)));
    }
```

7.3.2 Function resolveData

- TODO

```
    function resolveData(
        address addrRoot,
        uint256 id
    ) public view returns (address addrData) {
        TvmCell code = _buildDataCode(addrRoot);
        TvmCell state = _buildDataState(code, id);
        uint256 hashState = tvm.hash(state);
        addrData = address.makeAddrStd(0, hashState);
    }
```

7.4 Internal Method Definitions

7.4.1 Function _buildDataCode

- TODO

```
    function _buildDataCode(address addrRoot) internal virtual view
         returns (TvmCell) {
        TvmBuilder salt;
        salt.store(addrRoot);
        return tvm.setCodeSalt(_codeData, salt.toCell());
    }
```

7.4.2 Function _buildDataState

- TODO

```
    function _buildDataState(
        TvmCell code,
        uint256 id
    ) internal virtual pure returns (TvmCell) {
        return tvm.buildStateInit({
            contr: Data,
            varInit: {_id: id},
            code: code
        });
    }
```

Chapter 8

Contract Index

Contents

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

```
    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

```
    address _addrRoot;
```

```
    address _addrOwner;
```

8.5 Constructor Definitions

8.5.1 Constructor

Critical issue: Constructor for Index (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

```
    constructor(address root) public {
        optional(TvmCell) optSalt = tvm.codeSalt(tvm.code());
        require(optSalt.hasValue(), Errors.ERROR_EMPTY_SALT);
        (address addrRoot, address addrOwner) = optSalt
            .get()
            .toSlice()
            .decode(address, address);
        require(msg.sender == _addrData, Errors.
            ERROR_MESSAGE_SENDER_IS_NOT_OWNER);
        tvm.accept();
        _addrRoot = addrRoot;
        _addrOwner = addrOwner;
        if(addrRoot == address(0)) {
            _addrRoot = root;
        }
    }
```

8.6 Public Method Definitions

8.6.1 Function destruct

- TODO

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

8.6.2 Function getInfo

- TODO

```
    function getInfo() public view override returns (
        address addrRoot,
        address addrOwner,
        address addrData
    ) {
        addrRoot = _addrRoot;
        addrOwner = _addrOwner;
        addrData = _addrData;
    }
```

Chapter 9

Contract IndexBasis

Contents

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

```
    address static _addrRoot;
```

```
    uint256 static _codeHashData;
```

9.3 Modifier Definitions

9.3.1 Modifier onlyRoot

```
    modifier onlyRoot() {
        require(msg.sender == _addrRoot, Errors.
            ERROR_MESSAGE_SENDER_IS_NOT_ROOT);
        tvm.accept();
        _;
    }
```

9.4 Constructor Definitions

9.4.1 Constructor

Critical issue: Constructor for IndexBasis (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 loren ipsum

- TODO

```
    constructor() public onlyRoot {}
```

9.5 Public Method Definitions

9.5.1 Function destruct

- TODO

```
    function destruct() public onlyRoot {
        selfdestruct(_addrRoot);
    }
```

9.5.2 Function getInfo

- TODO

```
    function getInfo() public view returns (address addrRoot,
        uint256 codeHashData) {
        addrRoot = _addrRoot;
        codeHashData = _codeHashData;
    }
```

Chapter 10

Contract IndexResolver

Contents

10.1 Overview

In file `IndexResolver.sol`

10.2 Variable Definitions

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

```
    TvmCell _codeIndex;
```

10.3 Public Method Definitions

10.3.1 Function resolveCodeHashIndex

- TODO

```
    function resolveCodeHashIndex(
        address addrRoot,
        address addrOwner
    ) public view returns (uint256 codeHashIndex) {
        return tvm.hash(_buildIndexCode(addrRoot, addrOwner));
    }
```

10.3.2 Function resolveIndex

- TODO

```
    function resolveIndex(
        address addrRoot,
        address addrData,
        address addrOwner
    ) public view returns (address addrIndex) {
        TvmCell code = _buildIndexCode(addrRoot, addrOwner);
        TvmCell state = _buildIndexState(code, addrData);
        uint256 hashState = tvm.hash(state);
        addrIndex = address.makeAddrStd(0, hashState);
    }
```

10.4 Internal Method Definitions

10.4.1 Function _buildIndexCode

- TODO

```
    function _buildIndexCode(
        address addrRoot,
        address addrOwner
    ) internal virtual view returns (TvmCell) {
        TvmBuilder salt;
        salt.store(addrRoot);
        salt.store(addrOwner);
        return tvm.setCodeSalt(_codeIndex, salt.toCell());
    }
```

10.4.2 Function _buildIndexState

- TODO

```
    function _buildIndexState(
        TvmCell code,
        address addrData
    ) internal virtual pure returns (TvmCell) {
        return tvm.buildStateInit({
            contr: Index,
            varInit: {_addrData: addrData},
            code: code
        });
    }
```

Chapter 11

Contract Manager

Contents

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

```
    TvmCell _rootCode;
```

11.3 Constructor Definitions

11.3.1 Constructor

Critical issue: Constructor for Manager (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

```
    constructor (TvmCell rootCode) public {
        tvm.accept();
        _rootCode = rootCode;
    }
```

11.4 Public Method Definitions

11.4.1 Function deployRoot

- TODO

```
    function deployRoot(
        address addrOwner,
        TvmCell codeIndex,
        TvmCell codeData,
        string name,
        string description,
        string tokenCode,
        uint256 totalSupply,
        uint128 index,
        bytes part
    ) public view {
        tvm.accept();

        TvmCell stateNftRoot = _buildNftRootState(addrOwner);
        new NftRoot {stateInit: stateNftRoot, value: Constants.
            DEPLOY_INDEX_FEE}( codeIndex, codeData, name,
            description, tokenCode, totalSupply, index, part);
    }
```

11.5 Internal Method Definitions

11.5.1 Function _buildNftRootState

- TODO

```
    function _buildNftRootState( address addrOwner) internal
        virtual view returns (TvmCell) {
        TvmCell code = _rootCode.toSlice().loadRef();
        return tvm.buildStateInit({
            contr: NftRoot,
            varInit: {_addrOwner: addrOwner},
            code: code
        });
    }
```

Chapter 12

Contract NftRoot

Contents

12.1 Overview

In file `NftRoot.sol`

12.2 Contract Inheritance

DataResolver	
IndexResolver	

12.3 Static Variable Definitions

address	_addrOwner	

```
    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
		used in @2.NftRoot.getInfo
		used in @2.NftRoot.burn
		assigned in @2.Nft-Root.:constructor
		used in @2.NftRoot.:constructor

```
    uint256 _totalMinted;
```

```
    address _addrBasis;
```

```
    uint256 _totalSupply;
```

```
    string _name;
```

```
    string _description;
```

```
    string _tokenCode;
```

```
    mapping(uint128 => bytes) _content;
```

```
    uint128 _price;
```

12.5 Modifier Definitions

12.5.1 Modifier onlyOwner

```
    modifier onlyOwner() {
        require(msg.sender == _addrOwner, Errors.
            ERROR_MESSAGE_SENDER_IS_NOT_OWNER);
        tvm.accept();
        _;
    }
```

12.6 Constructor Definitions

12.6.1 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

```
    constructor(
        TvmCell codeIndex,
        TvmCell codeData,
        string name,
        string description,
        string tokenCode,
        uint256 totalSupply,
        uint128 index,
        bytes part
    ) public {
        tvm.accept();
        _codeIndex = codeIndex;
        _codeData = codeData;
        _name = name;
        _description = description;
        _tokenCode = tokenCode;
        _totalSupply = totalSupply;

        _content[index] = part;

        _price = 1 ton;
    }
```

12.7 Public Method Definitions

12.7.1 Function burn

- TODO

```
    function burn(address dataAddress, address owner) public
        onlyOwner {
        require(msg.value >= (_price), Errors.
            ERROR_MSG_VALUE_LESS_THAN_PRICE);

        Data(dataAddress).destruct
            {value: msg.value, flag: 3, bounce: true}
            (owner);
    }
```

12.7.2 Function deployBasis

- TODO

```
    function deployBasis(TvmCell codeIndexBasis) public onlyOwner {
        require(msg.value > 0.5 ton, Errors.ERROR_NOT_ENOUGH_GRAMS)
            ;
        uint256 codeHasData = resolveCodeHashData();
        TvmCell state = tvm.buildStateInit({
            contr: IndexBasis,
            varInit: {
                _codeHashData: codeHasData,
                _addrRoot: address(this)
            },
            code: codeIndexBasis
        });
        _addrBasis = new IndexBasis{stateInit: state, value: 0.4
            ton}();
    }
```

12.7.3 Function destructBasis

- TODO

```
    function destructBasis() public view onlyOwner {
        IIndexBasis(_addrBasis).destruct();
    }
```

12.7.4 Function getInfo

- TODO

```
    function getInfo() public view returns (
        mapping(uint128 => bytes) content,
        string name,
        string description,
        string tokenCode,
        uint256 totalSupply,
        uint128 price
    ) {
        content = _content;
        name = _name;
        description = _description;
        tokenCode = _tokenCode;
        totalSupply = _totalSupply;
        price = _price;
    }
```

12.7.5 Function mintNft

- TODO

```
    function mintNft(uint64 creationDate, string comment, address
        owner) public onlyOwner {
        require(msg.value >= 1.6 ton, Errors.ERROR_NOT_ENOUGH_GRAMS
            );
        require(_totalMinted <= _totalSupply, Errors.
            ERROR_MINTED_TOO_MUCH);
        TvmCell codeData = _buildDataCode(address(this));
        TvmCell stateData = _buildDataState(codeData, _totalMinted)
            ;
        new Data
            {stateInit: stateData, value: 1.5 ton} (
                owner,
                _codeIndex,
                msg.sender,
                _name,
                _description,
                _tokenCode,
                creationDate,
                comment,
                0,
                _content[0]
            );

        _totalMinted++;
    }
```

12.7.6 Function setPrice

- TODO

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