**Спецификация децентрализованной системы**

**INTDAO**

**Версия 1.0.0**

**Дата 21.07.2023 г.**

[Введение 5](#_Toc140856986)

[Общее описание системы и ее компонентов 5](#_Toc140856987)

[Oracle bot 5](#_Toc140856988)

[Auction bot 5](#_Toc140856989)

[Position bot 5](#_Toc140856990)

[Liquidity bot 5](#_Toc140856991)

[Arbitrage bot 5](#_Toc140856992)

[Описание работы смарт-контрактов 6](#_Toc140856993)

[INTDAO 6](#_Toc140856994)

[Общее описание 6](#_Toc140856995)

[Описание полей и структур данных 6](#_Toc140856996)

[Voting 6](#_Toc140856997)

[activeVoting 6](#_Toc140856998)

[Votings 6](#_Toc140856999)

[votingID 7](#_Toc140857000)

[votes 7](#_Toc140857001)

[ruleToken 7](#_Toc140857002)

[params 7](#_Toc140857003)

[addresses 7](#_Toc140857004)

[paused 7](#_Toc140857005)

[authorized 7](#_Toc140857006)

[pooled 8](#_Toc140857007)

[totalPooled 8](#_Toc140857008)

[NewVoting 8](#_Toc140857009)

[VotingSucceed 8](#_Toc140857010)

[VotingFailed 8](#_Toc140857011)

[Описание методов 8](#_Toc140857012)

[constructor 8](#_Toc140857013)

[setAddressOnce 10](#_Toc140857014)

[addVoting 11](#_Toc140857015)

[isEnoughTokensPooledToInitVoting 11](#_Toc140857016)

[poolTokens 11](#_Toc140857017)

[returnTokens 11](#_Toc140857018)

[vote 11](#_Toc140857019)

[claimToFinalizeCurrentVoting 11](#_Toc140857020)

[finalizeCurrentVoting 11](#_Toc140857021)

[receive 11](#_Toc140857022)

[withdraw 11](#_Toc140857023)

[CDP 12](#_Toc140857024)

[Общее описание 12](#_Toc140857025)

[Описание полей и структур данных 12](#_Toc140857026)

[Position 12](#_Toc140857027)

[numPositions 12](#_Toc140857028)

[dao 12](#_Toc140857029)

[oracle 12](#_Toc140857030)

[coin 13](#_Toc140857031)

[auction 13](#_Toc140857032)

[rule 13](#_Toc140857033)

[weth 13](#_Toc140857034)

[positions 13](#_Toc140857035)

[PositionOpened 13](#_Toc140857036)

[PositionUpdated 13](#_Toc140857037)

[markedOnLiquidation 13](#_Toc140857038)

[OnLiquidation 13](#_Toc140857039)

[Описание методов 14](#_Toc140857040)

[сonstructor 14](#_Toc140857041)

[renewContracts 14](#_Toc140857042)

[openCDP 14](#_Toc140857043)

[generatedFeeUnrecorded 14](#_Toc140857044)

[totalCurrentFee 14](#_Toc140857045)

[getMaxStableCoinsToMint 14](#_Toc140857046)

[getMaxStableCoinsToMintForPos 14](#_Toc140857047)

[claimInterest 14](#_Toc140857048)

[claimEmission(uint256 amount, address beneficiary) external 14](#_Toc140857049)

[closeCDP 14](#_Toc140857050)

[transferFee 14](#_Toc140857051)

[allowSurplusToAuction 14](#_Toc140857052)

[claimMarginCall 14](#_Toc140857053)

[startCoinsBuyOut 14](#_Toc140857054)

[finishMarginCall 14](#_Toc140857055)

[markToLiquidate 14](#_Toc140857056)

[eraseMarkToLiquidate 14](#_Toc140857057)

[updateCDP 14](#_Toc140857058)

[withdrawEther 14](#_Toc140857059)

[wethLocked 14](#_Toc140857060)

[isOnLiquidation 14](#_Toc140857061)

[burnRule 14](#_Toc140857062)

[receive 14](#_Toc140857063)

[withdraw 14](#_Toc140857064)

[Auction 15](#_Toc140857065)

[Cart 15](#_Toc140857066)

[Deposit 15](#_Toc140857067)

[Commodity 15](#_Toc140857068)

[exchangeRateContract 15](#_Toc140857069)

[InflationFund 15](#_Toc140857070)

[Rule 15](#_Toc140857071)

[stableCoin 15](#_Toc140857072)

[WETH 15](#_Toc140857073)

[Глоссарий 16](#_Toc140857074)

# Введение

Данный документ содержит описание системы INTDAO, ее функциональных возможностей и требований.

Документ может быть использован как разработчиками, аналитиками и тестировщиками, так и бизнесом для понимания механики работы системы.

# Общее описание системы и ее компонентов

INTDAO – это децентрализованная система, позволяющая производить эмиссию и обращение обеспеченного стэйблкоина TrueStableCoin (TSC). В качестве обеспечения используется нативная криптовалюта того блокчейна, на котором развертывается система. Все условия работы системы фиксируются в смарт-контрактах, описывая ее параметры и варианты использования. Таким образом, все условия работы и возможности по изменению функционала заранее зафиксированы и не могут быть изменены по воле какого-либо индивидуума или централизованного органа управления.

Система состоит из нескольких компонентов: набора смарт-контрактов INTDAO, веб-интерфейса для работы со смарт-контрактами (dApp), пулов ликвидности, бота для трансляции рыночных котировок в блокчейн (Oracle bot), бота для участия в аукционах, проводимых системой (Auction bot), бота по мониторингу позиций (Position bot), бота по поддержанию ликвидности заемной позиции (Liquidity bot), бота для арбитража в пулах (Arbitrage bot), скрипта последовательного развертывания контрактов.

Логика работы DAO основана на кредитной эмиссии стэйбл-коинов, обеспеченных ликвидным активом – базовой криптовалютой. Обеспечение кредита принимается с дисконтом, за пользование кредитом взымается ссудный процент, стоимость одного стэйбл-коина принимается равной одному доллару США, а затем колеблется в соответствии со стоимостью взвешенной корзины сырьевых инструментов, торгуемых на бирже. Полный список инструментов приводится в соответствующем разделе.

## Oracle bot

Парсит котировки из открытых или закрытых источников и транслирует их в блокчейн раз в заданный промежуток времени, либо по платному запросу пользователя.

## Auction bot

Следит за возникновением аукционов и делает ставку по близкой к рыночной цене. Старается выкупить залоги по максимально низкой цене.

## Position bot

Бот, просматривающий все позиции на предмет достаточности обеспечения, помечает их на ликвидацию и ликвидирует позиции в случае ее возможности.

## Liquidity bot

Следит за пользовательской заемной позицией и автоматически поддерживает достаточность обеспечения.

## Arbitrage bot

Арбитражит при возникновении дисбалансов в пулах относительно индикативной цены.

## Скрипт развертывания системы

Скрипт, который позволяет последовательно развернуть в сети все необходимые контракты с учетом заданной последовательности и необходимых адресов.

# Описание работы смарт-контрактов

Работа DAO описывается набором взаимосвязанных смарт-контрактов, каждый из которых покрывает по возможности изолированный функционал. Часть контрактов подразумевают их бесшовную замену путем проведения голосования акционеров INTDAO.

## INTDAO

### Общее описание

INTDAO – это ключевой контракт системы. В нем хранятся основные параметры всей системы, адреса, на которых развернуты другие контракты системы. Кроме этого, в данном контракте реализован функционал проведения собрания акционеров.

### Описание полей и структур данных

#### Voting

struct Voting

Описывает сущность *голосование*. Содержит следующие поля:

uint totalPositive – общее количество положительных голосов;  
uint256 voteingType – тип голосования.   
Может принимать следующие значения:   
1 – голосование за изменение параметра;

2 – голосование за изменение адреса контракта для взаимодействия;

3 – голосование за постановку/снятие какого-либо контракта на паузу;

4 – голосование за авторизацию контракта на осуществление критически важных для системы действий.  
string name – название переменной или адреса, за который проводится голосование;  
uint value – значение. Используется в случае голосований за параметр (тип 1);  
address payable addr – адрес. Используется при голосованиях за адрес (тип 2);  
uint256 startTime – дата и время начала голосования;  
bool decision – решение. Имеет значение да/нет, используется в голосованиях на авторизацию/деавторизацию контрактов в голосованиях на постановку/снятие контрактов на паузу;

#### activeVoting

bool public activeVoting

Флаг, отвечающий за наличие действующего голосования. Используется, чтобы одновременно могло проходить только одно голосование. Устанавливается в значение trueпри создании нового голосования и в false при завершении голосования с тем или иным результатом.

#### Votings

mapping (uint=>Voting) public votings

Структура данных для хранения всех голосований. По идентификатору голосования можно получить сущность *голосование*.

#### votingID

uint256 public votingID

Счетчик-идентификатор текущего голосования.

#### votes

mapping (uint => mapping(address => uint)) votes

Структура данных, хранящая для каждого голосования акционеров, голосовавших положительно и количество отданных положительных голосов.

#### ruleToken

Rule ruleToken

Экземпляр класса Rule – управляющих токенов. Используется для взаимодействия с акциями для их депонирования и голосования.

#### params

mapping (string => uint) public params

Структура данных, хранящая необходимые для работы параметры. По имени параметра можно получить его значение. В конструкторе устанавливаются первоначальные значения многих параметров, которые затем используются в работе INTDAO (см. constructor).

#### addresses

mapping (string => address payable) public addresses

Структура данных, хранящая необходимые для работы адреса. По имени адреса можно получить его значение. В конструкторе устанавливаются первоначальные значения некоторых адресов, которые затем используются в работе INTDAO (см. constructor).

#### paused

mapping (address => bool) public paused

Структура данных, хранящая адреса контрактов, которые находятся на паузе. По умолчанию, любой адрес не находится на паузе. В текущей реализации не используется.

#### authorized

mapping (address => bool) public authorized

Структура данных, хранящая адреса авторизованных контрактов, которые могут совершать те или иные критичные для системы действия (например, дополнительную эмиссию стэйбл-коинов, расходования стабилизационного фонда и т.п.).

По умолчанию авторизуется три контракта:

deposit (депозитный контракт), inflationFund (контракт для создания запрограммированной инфляции) и platform – контракт краудфандинговой платформы.

По решению собрания акционеров можно дополнительно авторизовать другие контракты, либо деавторизовать любой контракт.

#### pooled

mapping (address => uint) public pooled

Структура данных, хранящая количество задепонированных на контракте акций для каждого акционера. Депонирование используется во время проведения голосования с целью исключить возможность дважды проголосовать одной и той же акцией путем ее передачи на другой адрес.

#### totalPooled

uint256 public totalPooled

Счетчик суммарного количества задепонированных на данный момент акций для голосования. В случае, если токены депонируются через функцию pool, то данное значение должно совпадать с количеством акций на адресе контракта (rule.balanceOf(address(this))

#### NewVoting

event NewVoting (uint256 id, string name)

Событие, происходящее при создании нового голосования. Содержит идентификатор голосования и имя параметра/адреса, по которому проходит голосование.

#### VotingSucceed

event VotingSucceed (uint256 id)

Событие, происходящее при положительном решении по голосованию. Содержит идентификатор голосования.

#### VotingFailed

event VotingFailed (uint256 id)

Событие, происходящее при отрицательном решении по голосованию. Содержит идентификатор голосования.

### Описание методов

#### constructor

constructor (address WETH)

Конструктор вызывается при развертывании контракта в сети. Его единственная ответственность – инициализация начальных значений параметров и адресов.

В частности, устанавливаются следующие **параметры**:

interestRate – ссудный процент по кредиту (9%);

depositRate ­– процент по депозиту. Изначально устанавливается 8%;

liquidationFee – штраф, применяемый при ликвидации заемной позиции (margin call). Изначально устанавливается 13%;

collateralDiscount – дисконт, применяемый к залогу. Определяет максимальный размер кредита в стэйбл-коинах под предоставленное обеспечение. Изначально устанавливается 30%.

stabilizationFundPercent – количество стэйбл-коинов в стабилизационном фонде, хранящемся на контракте CDP. Определяется как процент от общего объема эмиссии стэйбл-коинов. Изначально устанавливается 5%.

quorum – кворум для принятия решения акционерами на голосовании. Определяется как процент от общего объема эмиссии акций (Rule). Изначально устанавливается 60%.

majority – процент большинства для принятия положительного решения по голосованию. Изначально устанавливается в 50%.

minCDPBalanceToInitBuyOut – минимальная сумма превышения стабилизационного фонда для инициирования обратного выкупа акций (Rule). Изначально устанавливается в 10 TSC.

absoluteMajority – абсолютное большинство при проведении голосования. Используется для досрочного прекращения голосования без ожидания истечения срока votingDuration. Изначально устанавливается в 80%.

minRuleTokensToInitVotingPercent – минимальный процент акций у акционера для инициирования голосования. Изначально устанавливается в 1%

votingDuration ­– продолжительность голосования в секундах. Используется при определении возможности принять положительное решение по голосованию. Изначально устанавливается 1 день (86 400 секунд).

auctionTurnDuration ­– время в секундах, отводимое участникам различных аукционов для своей ставки с момента последней ставки. Изначально устанавливается 15 минут (900 секунд).

minAuctionPriceMove – минимальный шаг аукциона в процентах от последней ставки. Изначально устанавливается в 5%.

minColleteral – минимальное обеспечение по кредиту в абсолютном значении. Изначально устанавливается в 0.01 ETH.

marginCallTimeLimit – время в секундах, отводимое владельцу заемной позиции на то, чтобы увеличить обеспечение или сократить эмиссию (вернуть часть кредита). Изначально устанавливается 1 день (86 400 секунд).

annualInflationPercent – процент инфляции. Возможный объем необеспеченной эмиссии стэйбл-коинов TSC, производимой раз в год. Изначально устанавливается 1%.

defaultDepositPeriod – период, на который по умолчанию размещается депозит. процент инфляции

maxCoinsForStabilization – максимальная сумма TSC, которую единовременно можно запросить для пополнения стабилизационного фонда при допэмиссии акций (Rule). Изначально устанавливается в 50 TSC.

maxRuleEmissionPercent ­­– максимальный процент акций, который может быть выпущен при проведении аукциона для пополнения стабилизационного фонда. Изначально устанавливается в 1%. Определяется на этапе принятия ставки по аукциону (makeBid).

highVolatilityEventBarrierPercent – изменения котировок в процентном соотношении, которое приводит к событию highVolatility по тому или иному инструменту в контракте exchangeRateContract.

minCoinsToMint – минимальное количество стэйблкоинов, возможное для выпуска при открытии позиции.

Кроме параметров, в конструкторе устанавливаются **адреса** используемых в системе контрактов:

weth – адрес контракта обернутого эфира. Берется из входных параметров конструктора. Обычно в популярных сетях (ETH, ETC) этот контракт уже кем-то развернут, и нужно просто вручную установить его адрес;

cdp – адрес кредитного контракта;

auction – адрес депозитного контракта;

stableCoin – адрес стэйбл-коина TSC;

dao – адрес контракта INTDAO;

oracle – адрес контракта для получения котировок криптовалюты и биржевых инструментов;

rule – адрес контракта акций (управляющих токенов Rule);

deposit – адрес депозитного контракта;

inflationFund – адрес контракта, определяющего порядок инфляционной эмиссии;

inflationSpender – адрес контракта, бенефициара инфляционной эмиссии. Изначально, бенефициаром устанавливается кредитный контракт, то есть при инфляционной эмиссии будет пополняться стабилизационный фонд системы. В дальнейшем предполагается, что это будет адрес фонда различных благотворительных организаций.

#### setAddressOnce

function setAddressOnce(string memory addressName, address payable addr) public

addressName – имя адреса, который необходимо установить, addr – значение адреса.

Функция используется для единократной установки некоторого адреса в определенное значение. Фактически используется только после развертывания всех контрактов в системе. INTDAO должно знать адреса контрактов для взаимодействия, но в момент развертывания самого INTDAO они отсутствую, поэтому устанавливаются единожды по мере развертывания необходимых контрактов (например, CDP).

#### addVoting

function addVoting(uint256 votingType, string memory name, uint value, address payable addr, bool \_decision) public

votingType тип голосования, name имя параметра или адреса, который требуется поменять, value значение (используется при голосовании типа 1 за значение параметра), addr адрес (используется при голосовании за адрес какого-либо контракта, тип голосования – 2), \_decision – решение (да/нет при голосованиях типа 3 за установку/снятие паузы на работу каких-либо контрактов или методов, а также голосованиях типа 4 на авторизацию/деавторизацию каких-либо контрактов). Ничего не возвращает.

если нет текущего активного голосования, votingType попадает в диапазон от 1 до 4 и пользователь обладает необходимым количеством задепонированных на контракте акций для создания голосования (minRuleTokensToInitVotingPercent), то создать новое голосование с заданными параметрами.

#### isEnoughTokensPooledToInitVoting

function isEnoughTokensPooledToInitVoting(address initiator) internal view returns (bool enough)

initiator – адрес пользователя, который пытается создать голосование. Возвращает да/нет.

Внутренняя функция, проверяет, достаточно ли пользователь задепонировал акций (Rule) для создания нового голосования, сравнивая его с процентом (minRuleTokensToInitVotingPercent) от общего объема эмиссии акций (Rule). Этот функционал помогает ограничивать от спама на создание новых голосований пользователями с минимальным количеством акций.

#### poolTokens

function poolTokens() public returns (bool success)

Положить свои акции на депозит. Проверяет объем акций Rule, которые пользователь разрешил расходовать контракту. Затем переводит весь разрешенный объем на адрес контракта INTDAO, увеличивает значения переменных totalPooled и задепонированных акций для конкретного пользователя (pooled[msg.sender]). Возвращает true в случае успеха.

#### returnTokens

function returnTokens() public returns (bool)

Метод возвращает все токены, которые пользователь ранее задепонировал для голосования. totalPooled при этом уменьшается, pooled[msg.sender] обнуляется.

При наличии активного голосования и условия, что пользователь проголосовал положительно, уменьшается значение totalPositive у текущего голосования.

#### **vote**

function vote(bool \_vote) public

\_vote – решение акционера по текущему голосованию. Возможные значения – за/против.

Проверка наличия текущего активного голосования, не истечения срока по нему, наличия у пользователя задепонированных акций Rule, что автоматически означает, что данный пользователь является акционером.

Если пользователь голосует «за», проверяем, голосовал ли этот же пользователь по текущему голосованию «за» до этого. Если да, то к общему количеству положительных голосов добавляется только разница. Единственный кейс, когда это возможно – пользователь задепонировал часть своих акций, затем проголосовал «за», задепонировал еще акций и проголосовал «за» повторно. Разница между задепонированными пользователем акциями и его положительными голосами по текущему голосованию может быть только больше нуля, потому что уменьшить количество акций на депозите частично нельзя. Можно снять только все акции сразу.

Если пользователь голосует «против», то проверяется, не голосовал ли он раньше меньшим или таким же количеством голосов «за» по текущему голосованию. Если голосовал, мы уменьшаем общее количество голосов «за» на количество ранее отданных этим пользователем голосов «за», обнуляем его количество голосов «за» по текущему голосованию.

#### claimToFinalizeCurrentVoting

function claimToFinalizeCurrentVoting() public

Публичная функция, делающая попытку завершить текущее голосование. Проверка наличия действующего голосования. Если удовлетворяется условие абсолютного большинства, завершает голосование без проверки истечения его срока. В противном случае, проверяем, прошел ли срок проведения голосования (votingDuration). Если да, то проверяем сперва наличие кворума для принятия решения, и большинства голосов «за». Если это условие выполняется, завершаем голосование положительно. Если срок вышел, но кворума нет, либо есть кворум, но нет достаточного количества голосов «за» – завершаем голосование отрицательно (принимается решение против) и выстреливается соответствующее событие VotingFailed с указанием идентификатора текущего голосования и флаг наличия действующего голосования activeVoting устанавливается в false.

#### finalizeCurrentVoting

function finalizeCurrentVoting() internal

Внутренняя функция, используется только изнутри самого контракта. Вызывается из claimToFinalizeCurrentVoting при условии возможности принять положительное решение по голосованию. Определяется тип голосования (1-4), и необходимое значение устанавливается в то, которое было предложено на голосовании. Выстреливается событие VotingSucceed с указанием идентификатора текущего голосования. Флаг наличия действующего голосования activeVoting устанавливается в false.

#### receive

receive() external payable

Если кто-либо случайно отправит эфир на адрес контракта, это будет восприниматься как благотворительное пожертвование. Криптовалюта пойдет на поддержание работы системы – на контракт по трансляции котировок.

#### withdraw

function withdraw() public

Вывод все баланса криптовалюты с контракта на адресу оракула.

## CDP

### Общее описание

Контракт CDP – от Collateral Debt Position предоставляет функционал для создания, хранения и обновления заемных позиций (кредитов) в стэйбл-коинах.

### Описание полей и структур данных

#### Position

struct Position

Описывает сущность *заемная позиция*. Содержит следующие поля:

address owner – владелец позиции;  
uint256 coinsMinted – количество выпущенных в кредит стэйбл-коинов в данной позиции;  
uint256 wethAmountLocked – объем обернутого эфира, принятого в обеспечение;  
uint256 feeGeneratedRecorded;  
uint256 timeOpened – дата и время открытия позиции;  
uint256 lastTimeUpdated – дата и время последнего обновления позиции;  
uint256 interestRate – ссудный процент, под который получены стейбл-коины;  
uint256 markedOnLiquidationTimestamp – время, когда позиция была помечена на ликвидацию, означает, что позиция будет в ближайшее время раскрыта и залог будет реализован через аукцион;bool onLiquidation – метка, означающая, что позиция находится в стадии ликвидации, обеспечение реализуется на аукционе;  
bool liquidated – метка, означающая, что позиция ликвидирована (залог продан);  
uint256 liquidationAuctionID – идентификатор аукциона, через который был реализован залог;  
bool restrictInterestWithdrawal – если флаг поднят, то только самому пользователю разрешается оплачивать проценты по кредиту (вызывать метод transferInterest). В противном случае, метод transferInterest может быть вызван любым пользователем;

#### numPositions

uint256 public numPositions

Счетчик позиций – хранит общее количество заемных позиции в INTDAO.

#### dao

INTDAO dao

Экземпляр класса INTDAO. Используется для получения параметров и адресов системы.

#### oracleCart

cartContract oracleCart

Экземпляр класса cartContract. Используется для получения необходимых рыночных котировок.

#### coin

stableCoin coin

Экземпляр класса stableCoin – используется для взаимодействия с контрактом stableCoin: выпуска, уничтожения, контроля переводов и балансов стейбл-коинов, в том числе балансов контрактов.

#### auction

Auction auction

Экземпляр класса Auction – для инициации и контроля хода выполнения аукционов.

#### **rule**

Rule rule

Экземпляр класса Rule – управляющих токенов (акций INTDAO).

#### **weth**

ERC20 weth

Экземпляр класса weth – обернутого эфира. Так как для правильной работы CDP необходимо, чтобы залог обладал интерфейсом ERC20, в качестве залога хранится обернутый эфир.

#### positions

mapping(uint256 => Position) public positions

Структура данных, позволяющая по идентификатору позиции получить структуру *заемная позиция.*

#### **PositionOpened**

event PositionOpened (address owner, uint256 posId)

Событие, происходящее, когда открывается новая *заемная позиция*. Содержит поля *владелец позиции* и *идентификатор позиции*.

#### **PositionUpdated**

event PositionUpdated (uint256 posID, uint256 newStableCoinsAmount, uint256 wethLocked)

Событие, происходящее, когда *заемная позиция* обновляется. Содержит поля идентификатор позиции, обновленное количество стэйблкоинов TSC, выпущенное в рамках данной позиции, объем обеспечения.

#### **markedOnLiquidation**

event markedOnLiquidation (uint256 posID, uint256 timestamp)

Событие, означающее, что позиция с заданным идентификатором помечена на ликвидацию в указанное время.

#### markOnLiquidationErased

event markedOnLiquidationErased (uint256 posID, uint256 timestamp)

Событие, означающее, что пометка на ликвидацию позиции с заданным идентификатором снята в указанное время.

#### **OnLiquidation**

event OnLiquidation (uint256 posID, uint256 timestamp);

Событие, означающее, что позиция с заданным идентификатором находится на ликвидации с указанного времени.

### Описание методов

#### сonstructor

constructor(address payable INTDAOaddress)

Инициирует dao по INTDAOaddress, устанавливает адрес своего экземпляра в dao, задает себя же как inflationSpender. Инициализирует экземпляры контрактов coin, oracle, auction, rule и weth по адресам из dao. Эти адреса должны быть уже определены на момент развертывания контракта cdp согласно скрипту деплоя.

#### renewContracts

function renewContracts() public

Метод позволяет обновить экземпляры контрактов coin, oracle, auction, rule и weth по адресам из dao. Используется, в случае, если какой-либо из адресов контрактов был заменен путем голосования

#### openCDP

function openCDP(uint StableCoinsToMint) external payable returns (uint256 posID)

StableCoinsToMint – количество стэйблкоинов TSC, запрашиваемое в кредит. Функция подразумевает перевод криптовалюты (payable). В случае успеха, возвращает идентификатор новой позиции posID.

Проверяет максимальное количество стэйблкоинов, которое можно выпустить под предоставляемый залог. Если запрашиваемое количество больше – выпускается максимально возможный объем. Если запрашивается выпуск меньшего количества стэйблкоинов, чем максимально возможное, то выпускается запрашиваемое количество стэйблкоинов. Проверяется, что выпускается

#### interestAmountUnrecorded

function interestAmountUnrecorded(uint256 posID) public view returns (uint256 interestAmount)

posID – идентификатор позиции, возвращает объем процентов.

Только читает данные, ничего не модифицирует (view). Используется, чтобы получить текущее количество процентов по позиции, которые еще не были зафиксированы в блокчейн с последнего обновления позиции.

#### totalCurrentFee

function totalCurrentFee(uint256 posID) public view returns (uint256 fee)

posID – идентификатор позиции, возвращает объем процентов.

Только читает данные, ничего не модифицирует (view). Используется, чтобы получить общее количество процентов по позиции, как записанных, так и начисленных с последнего обновления позиции.

#### getMaxStableCoinsToMint

function getMaxStableCoinsToMint(uint256 ethValue) public view returns (uint256 amount)

На входе ethValue – объем обеспечения, на выходе – максимально возможное количество стейбл-коинов, которые контракт может выпустить под это обеспечение выпустить.

Берет из контракта cart текущую котировку stb, то есть стоимость эфира, поделенную на стоимость корзины. При постоянном значении стоимости криптовалюты если корзина подорожала, значение stb меньше, если подешевела – больше (см. контракт cart). Дальше определяется, какое максимальное количество стэйбл-коинов может быть выпущено с учетом дисконта, принятого в INTDAO.

#### getMaxStableCoinsToMintForPos

function getMaxStableCoinsToMintForPos(uint256 posID) public view returns (uint256 maxAmount)

На входе – идентификатор позиции, на выходе – максимально возможное количество стейбл-коинов, которые контракт может выпустить для этой позиции.

Из максимального количества монет с учетом обеспечения для текущей позиции вычитается количество процентов, начисленных по текущей позиции.

#### claimInterest

function claimInterest(uint256 amount, address beneficiary) external

amount – сколько, beneficiary – кому.

Позволяет исключительно авторизованным в INTDAO контрактам переводить средства требуемым пользователям со счета CDP, то есть из стабилизационного фонда. В частности, используется в контракте Deposit для начисления процентов владельцам депозитов. Если в стабилизационном фонде достаточно средств, то происходит их перевод. Если средств недостаточно, то переводятся все доступные средства, а на оставшуюся сумму, требуемую пользователю, устанавливается разрешение на расходование стэйблкоинов из стабилизационного фонда по мере их появления.

#### claimEmission(uint256 amount, address beneficiary) external

amount – сколько допечатать, beneficiary – для кого.

Функция позволяет авторизованным контрактам из массива dao.authorized производить дополнительную эмиссию стэйблкоинов. В данной реализации используется только контрактом InflatioinFund при инфляционной эмиссии.

#### closeCDP

function closeCDP(uint256 posID) external

Закрытие заемной позиции по идентификатору.

Проверяем, что позицию пытается закрыть ее владелец. Проверка того, что позиция не находится в стадии ликвидации. Попытка перевести с адреса владельца на адрес контракта CDP полную сумму долга (основной долг + проценты). Предварительно, пользователь должен разрешить расходование полной суммы долга со своего счета и обеспечить необходимый баланс. Если возврат долга произошел, владельцу позиции переводится его обеспечение и обнуляется значение обеспечения по кредиту. Затем с адреса контракта CDP происходит сжигание основной суммы долга и устанавливается флаг, что позиция ликвидирована (liquidated).

#### transferInterest

function transferInterest(uint256 posID) external

Внешняя функция, входной параметр – идентификатор позиции.

Если у данный позиции стоит флаг restrictInterestWithdrawal, производим проверку, что она вызывается владельцем позиции и если ее вызывает другой пользователь, делаем откат транзакции. Проверяем, что позиция не находится в стадии ликвидации (onLiquidation). Пытаемся перевести полные проценты по заемной позиции со счета пользователя на счет контракта. Предварительно пользователь должен разрешить расходование. Обнуляем записанный процент по заемной позиции (interestAmountRecorded) и обновляем время обновления позиции (lastTimeUpdated).

#### setRestrictInterestWithdrawal

function setRestrictInterestWithdrawal(uint256 posID, bool to) external

Идентификатор позиции, значение to, в которое нужно установить флаг restrictInterestWithdrawal.

Проверяем, что метод вызывается владельцем позиции. Если так – устанавливаем флаг restrictInterestWithdrawal в указанное значение to.

#### allowSurplusToAuction

function allowSurplusToAuction() external

Метод используется для предоставления разрешения на расходование средств с адреса контракта CDP контрактом Auction. Его вызов оправдан тогда, когда стабилизационный фонд, то есть баланс контракта в стэйбл-коинах, превышает заданный в INTDAO процент от общей эмиссии стэйбл-коинов (stabilizationFundPercent). Если эти условия выполняются и излишек стабилизационного фонда превышает значение minCDPBalanceToInitBuyOut, то есть минимальной суммы для инициирования обратного выкупа акций Rule, разрешаем контракту Auction расходование излишка.

#### claimMarginCall

function claimMarginCall(uint256 posID) external returns (bool success)

Функция используется для раскрытия позиции и ликвидации залога. В первую очередь проверяем, что позиция уже была ранее помечена на ликвидацию (markedOnLiquidationTimestamp) и время marginCallTimeLimit, отводимое пользователю на предоставление достаточности обеспечения истекло, а также что позиция не ликвидирована и не находится в стадии ликвидации. Если обеспечения все еще не хватает, помечаем, что позиция перешла в статус «на ликвидации», увеличиваем право расходования обернутого эфира с адреса CDP адресу Auction и устанавливаем флаг onLiquidation, выстреливаем событие OnLiquidation и возвращаем true. Если обеспечение в этот момент оказалось достаточным, обнуляем метку markedOnLiquidationTimestamp и возвращаем false.

#### startCoinsBuyOut

function startCoinsBuyOut(uint256 posID) external

Начать обратный выкуп стэйбл-коинов (то есть ликвидацию позиции). Если позиция на ликвидации и еще не ликвидирована, вызываем метод initCoinsBuyOut в контракте auction и обнуляем сумму залога (так как в методе initCoinsBuyOut залог должен быть переведен с контракта CDP на контракт auction. Метод initCoinsBuyOut возвращает идентификатор аукциона, в рамках которого будет реализован залог. Это значение присваивается полю liquidationAuctionID в выбранной позиции, чтобы затем можно было отследить, через какой аукцион реализовывался залог.

#### finishMarginCall

function finishMarginCall(uint256 posID) external

#### markToLiquidate

#### eraseMarkToLiquidate

#### updateCDP

#### withdrawEther

#### wethLocked

#### isOnLiquidation

#### burnRule

#### receive

#### withdraw

## Auction

## Cart

## Deposit

## Commodity

## exchangeRateContract

## InflationFund

## Rule

## stableCoin

## WETH

# Глоссарий

В глоссарии собраны термины, трактовка которых валидна в рамках описания системы INTDAO.

**DAO** – от англ. Decentralized Autonomous Organization, децентрализованная автономная организация. Это информационная система, основанная на смарт-контрактах, все правила в которой жестко зафиксированы в программном коде, хранящемся и исполняемом в блокчейн. Смарт-контракт для DAO является аналогом устава юридического лица.

**Ethereum** –программная платформа для создания блокчейнов (ETH, ETC, BSC, Matic и т.д.) и одноименный блокчейн (Ethereum mainnet).

**Блокчейн** ­– децентрализованная система, хранящая распределенный реестр адресов и транзакций, содержащая алгоритм построения новых блоков, и, возможно, среду исполнения программного кода.

**Токен** – учетная единица в блокчейн, порядок эмиссии и обращения которой определяется смарт-контрактом.

**ERC20** – популярный стандарт, описывающий интерфейс взаимозаменяемых токенов.

**Смарт-контракт** – это программный код, написанный на языке Solidity. После написания и проверки он может быть развернут (установлен) в тестовый или боевой блокчейн. После развертывания в боевом блокчейне, смарт-контракт получает уникальный адрес и программный код смарт-контракта не может быть изменен.