

Version	Description of Change	Author	Date
0.1	Draft	Richard	

CONTENTS

Summary	2
Credit token contract (ALAKEMI)	3
Messaging contract (AlkemiMessaging)	5
MAIN FUNCTIONS AT THE MESSAGING CONTRACT “ALKEMIMESSAGING”	7
Demo Script	14
ENVIRONAMNT DESCRIOPTION	14
CREDIT TOKEN CONTRACT (ERC20)	14
Check the Contract Symbol Name	14
Check the contract Owner	14
Check the contract owner token balance (By default, the owner own all 1 billion tokens)	14
MESSAGING CONTRACT (ALKEMIMESSAGING)	15
Check the “COST” as specified at the contract	15
Check Credit purchase base price	15
Check the specified credit amount assigned to new address upon they are authorized	15
Switch to Owner account and deposit 10000 tokens from ERC20 credit token contract to messaging contract	16
Specified owner address and check 1000 tokens are deposited to owner account successfully via following function at the messaging contract	17
Authorize a new address by owner	18
Add a new message / Remove a message	22
Buy Credits	25
Deposit credits from “ALAKEMI” ERC20 smart contract to messaing contract “AlkemiMessaging”	28
Withdraw credits from messaging contract “AlkemiMessaging” to “ALAKEMI” ERC20 smart contract	30

Summary

The smart contract design as specified at this document is based on following requirement:

Hi Richard,

We are sending a test out to our applicants to see if they have solid foundation in smart contracts. If you are interested in working with us please take a look at the below problem.

"Create a smart contract that facilitates the storage of messages on-chain. The smart contract owner should be able to authorize new addresses to store messages on the chain. Each newly added user will have a certain credit amount that they are assigned on add. From there each time they add or remove a message they should lose a credit. Users who are authorized should be able to purchase extra credits by sending funds to the smart contract. If a user is done they should be able to withdrawal any remaining credit funds.

There will be 2 smart contracts created based on this requirement, those two smart contracts are for “demo” only, there still have lots of space to improve (such as separate business logic from messaging contract to a library contract etc)

In order to build secure and robust contracts, Openzeppelin and best practices are followed at those two smart contracts (such as: safemath to avoid overflow, ReentrancyGuard to avoid the re-entrancy issue etc).

Credit token contract (ALAKEMI)

This is an ERC20 token with name as “ALAKEMI”. This smart contract will be used as “Credits” as specified at the requirement. And it implemented all interfaces as defined at ERC20 standard.

The specific wording of those function is followed by a clarification of what it does, in square brackets.

1. totalSupply [Get the total token supply]
2. balanceOf [Get the account balance of another account with address _owner]
3. transfer [Send _value amount of tokens to address _to]
4. transferFrom [Send _value amount of tokens from address _from to address _to]
5. approve [Allow _spender to withdraw from your account, multiple times, up to the _value amount. If this function is called again it overwrites the current allowance with _value]
6. allowance [Returns the amount which _spender is still allowed to withdraw from _owner]

Events format:

1. Transfer [Triggered when tokens are transferred.]
2. Approval

▼ ALAKEMI at 0xbbf...732db (memory) 

addPauser	address account	▼
approve	address spender, uint256 value	▼
pause		
renouncePauser		
transfer	address to, uint256 value	▼
transferFrom	address from, address to, uint256 value	▼
unpause		
allowance	address _owner, address spender	▼
balanceOf	address _owner	▼
0: uint256: 338		
isPauser	address account	▼
name		
owner		
paused		
symbol		
totalSupply		

Messaging contract (AlkemiMessaging)

The main functions of this contract are:

- Transfer credit tokens between this smart contract(messaging contract) and credit token (ERC20) contract
- (Owner Only) Authorized new player and assign credits to new player
- Create new message by authorized players, certain amount of tokens will be withdraw from current player account after the message adding
- Remove existing message. Only message owner can remove the message and certain amount of credits will be withdraw from current player account after the message removing
- Authorized players has the ability to buy credits using Ether
- Function to Check player details. Such as: player credit balance, is Authorized?
- Function to Check global credit status. Such as: total number of authorized player, total assigned credits, total raised credits, total raised Ether, total remaining credits etc.
- Check message details based on message ID. Such as: message creation date/time, message creator, message content and remove flag etc

Other Functions are shared by above two contracts:

- Pause - Only contract owner has the ability to pause the contract. After the contract is paused, all transactions will be denied (such as those smart contracts are in maintenance, we need pause them)
- Unpause – Contract owner has the ability to unpause the contract
- AddPausr – Contract owner has the ability to add another address as contract administrator so that those address also have the ability to pause/unpause contract

AlkemiMessaging at 0xdc5...79154 (memory)	
(fallback)	
addPauser	address account
authorizeNewPlayer	address _newplayer
buyCredits	
createMessage	string _messagecontent
depositCredits	uint256 _amount
kill	
pause	
removeMessage	uint256 _messageID
renouncePauser	
unpause	
withdrawCredits	uint256 _amount
COST	
CREDITPRICE	
CREDITSONAUTHORIZED	
getCreditStatus	
getMessageInfo	uint256 _messageid
getPlayerInfo	address _player
isPauser	address account

Main functions at the messaging contract “ALKEMIMESSAGING”

Authorize New Player

```
authorizeNewPlayer(address _newplayer) external onlyAdministrators nonReentrant  
isHuman whenNotPaused payable {
```

Input(s)

_newplayer

New player address

Return value

None

Function Details

- Modifier
 - Only contract owner can authorized a new player
 - Any call from any smart contract will be denied to avoid any attack from those smart contract
 - Use “ReentrancyGuard” and make sure Reentrancy not happens
 - If the contract is paused by owner, this function call will be denied
- Validation checking
 - New player address cannot be blank
 - The specified new player address is not authorized yet
 - Make sure there is enough credit tokens in order to assign credits to new player
- Transfer assigned credits from owner account to new player account
- Update global credit status: assignedCreditTokens , authorizedPlayers and remainingCreditTokens
- Update player authorization flag

```
//Authorized a new player (Owner only / Not allow Reentrancy / block contract call / Contract is not paused)  
function authorizeNewPlayer(address _newplayer) external onlyAdministrators nonReentrant isHuman whenNotPaused payable {  
    require(_newplayer != address(0),"Please specify new player address");  
    require(player[_newplayer].isAuthorized == 0,"This player already was authorized");  
    require(creditStatus.remainingCreditTokens > CREDITSONAUTHORIZED , "Run out of credits, Please contact owner");  
    require(player[owner].creditBalance >= CREDITSONAUTHORIZED , "Owner don't have enough balance");  
  
    //Transfer credit tokens from owner account to new player account at this contract  
    player[owner].creditBalance = player[owner].creditBalance.sub(CREDITSONAUTHORIZED);  
    player[_newplayer].creditBalance = player[_newplayer].creditBalance.add(CREDITSONAUTHORIZED);  
    //Update global credit status  
    creditStatus.remainingCreditTokens = creditStatus.remainingCreditTokens.sub(CREDITSONAUTHORIZED);  
    creditStatus.assignedCreditTokens = creditStatus.assignedCreditTokens.add(CREDITSONAUTHORIZED);  
    creditStatus.authorizedPlayers++;  
  
    player[_newplayer].isAuthorized = 1;    //Assign 1 to authorized new player.  
  
    emit evt_AuthorizedNewPlayer(msg.sender,_newplayer);  
}
```

Add new message

```
function createMessage(string _messagecontent) external whenNotPaused isHuman  
nonReentrant returns (uint256)
```

Input(s)

_messagecount

Message content

Return value

After new message is created, it generates a message ID for that new message. That New added message ID will be returned

Function Details

- Modifier
 - Any call from any smart contract will be denied to avoid any attack from those smart contract
 - Use “ReentrancyGuard” and make sure Reentrancy not happens
 - If the contract is paused by owner, this function call will be denied
- Validation checking
 - Only authorized player can add a new message
 - Have enough credit tokens in order to add a new message
- Withdraw certain amount of credits from current player account
- Update credit global status: arisedCredits
- Record new message to the chain: messageID, creation date/time, creator and message content

```
}
```

```
/* ======Messaging facilitates \ Create new Message (Not allow Reentrancy / block contract call / Contract is not paused)=====
```

```
function createMessage(string _messagecontent) external whenNotPaused isHuman nonReentrant returns (uint256) {  
    require(player[msg.sender].isAuthorized == 1,"This player was not authorized yet");  
    require(player[msg.sender].creditBalance >= COST , "You don't have enough credit tokens to add a new message");  
  
    messageCount++;  
  
    player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(COST); //Pay the credit before adding the message  
  
    creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST); //Update global credit status  
  
    //Record added message info  
    messageinfo[messageCount].CreateDate = now;  
    messageinfo[messageCount].Creator = msg.sender;  
    messageinfo[messageCount].MessageContent =_messagecontent;  
  
    emit evt_MessageAction(msg.sender, messageCount, "Add");  
    return messageCount; //return added message ID  
}
```

Remove message

```
removeMessage(uint256 _messageID) external whenNotPaused isHuman nonReentrant
returns (bool)
```

Input(s)

_ messageID
Message ID

Return value

true or false

Function Details

- Modifier
 - Any call from any smart contract will be denied to avoid any attack from those smart contract
 - Use “ReentrancyGuard” and make sure Reentrancy not happens
 - If the contract is paused by owner, this function call will be denied
- Validation checking
 - Make sure the specified message is NOT removed yet
 - Make sure current player is the specified message creator (only message creator can remove their own message)
 - Make sure the input message ID is a valid ID
 - Make sure current player is authorized
 - Have enough credit tokens in order to add a new message
- Withdraw certain amount of credits from current player account
- Update credit global status: arisedCredits
- Update the specified message and leave the remove flag

```
/* ===== Messaging facilitates \ Remove an existing Message(Not allow Reentrancy / block contract call / Contract is not paused
function removeMessage(uint256 _messageID) external whenNotPaused isHuman nonReentrant returns (bool) {
    require(_messageID <= messageCount, "Invalid Message ID");
    require(player[msg.sender].isAuthorized == 1, "This player was not authorized yet");
    require(messageinfo[_messageID].isRemoved == false, "This message already removed");
    require(player[msg.sender].creditBalance > COST, "You don't have no enough credit tokens to remove a new message");
    require(messageinfo[messageCount].Creator == msg.sender, "Only message creator can remove this message");

    player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(COST);      //Pay the credit before adding the message
    creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST); //Update global credit status
    messageinfo[messageCount].isRemoved = true;   //Leave message remove flag
    emit evt_MessageAction(msg.sender, _messageID, "Remove");
    return true;
}
```

Deposit Credits

```
depositCredits(uint _amount) external whenNotPaused nonReentrant payable returns  
(bool)
```

Input(s)

_amount

Return value

true or false

Function Details

- Transfer specified credit amount for current address from ERC20 credit smart contract to current smart contract

```
/* =====Credit tokens Operation \ Deposit credit token from ERC20 contract to this contract===== */  
function depositCredits(uint _amount) external whenNotPaused nonReentrant payable returns (bool){  
    require(creditTokens.transferFrom(msg.sender, address(this), _amount) == true); //Transfer credit tokens from current player &  
    player[msg.sender].creditBalance = player[msg.sender].creditBalance.add(_amount); //Record the transferred amounts to current  
    emit evt_TokenAction(msg.sender,_amount, "Deposit");  
}  
}
```

Withdraw Credits

```
withdrawCredits(uint _amount) external whenNotPaused nonReentrant payable returns  
(bool)
```

Input(s)

_amount

Return value

true or false

Function Details

- Transfer specified credit amount for current address from messaging smart contract to ERC20 credit smart contract.

```
/* =====Credit tokens Operation \ Withdraw credit token from current contract to ERC20 contract===== */  
function withdrawCredits(uint _amount) external whenNotPaused nonReentrant payable returns (bool){  
    require(player[msg.sender].creditBalance >= _amount,"You don't have enough credits");  
    require(creditTokens.transfer(msg.sender, _amount) == true); //Transfer credit tokens from current contract to this player &  
    player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(_amount); //remove transferred amount from current pl  
    emit evt_TokenAction(msg.sender,_amount, "Withdraw");  
}
```

Buy Credits

```
buyCredits() external nonReentrant isHuman whenNotPaused payable
```

Input(s)

Ether amount

Return value

None

Function Details

- Modifier
 - Block the call from smart contract to avoid any attack from other contract
 - Use “ReentrancyGuard” and make sure Reentrancy not happens
 - If the contract is paused by owner, this function call will be denied
- Validation checking
 - Make sure current address is authorized
 - Make sure the Ether amount is greater than zero
 - Make sure there are still some remaining credits
- Calculate purchased credit amount based on Ether as well as base price and make sure there are enough remaining credits amount
- Transfer the Ether to the contract owner account
- Update Credits global status: arised Ether ad remainingCredits
- Transfer the purchased credits amount from owner account to current address at the Credit ERC20 Token contract

```
/* =====Credit tokens Operation \ Buy credit tokens using Ether (Wei)===== */
function buyCredits() external nonReentrant isHuman whenNotPaused payable {
    require(player[msg.sender].isAuthorized == 1,"This player was not authorized yet");
    require(msg.value > 0,"Ether amount is zero");
    require(creditStatus.remainingCreditTokens > 0 , "All credits sold out");

    uint256 weiAmount = msg.value; //Ether in Wei amount
    uint256 credits = weiAmount.mul(CREDITPRICE); // Calculate credit tokens to sell based on base price

    require(credits <= creditStatus.remainingCreditTokens,"No enough remaining credit tokens");

    owner.transfer(weiAmount); // Send Eth from buyer to contract owner before any status changes

    //Update Credit global status
    creditStatus.arisedEth = creditStatus.arisedEth.add(weiAmount); //
    creditStatus.remainingCreditTokens = creditStatus.remainingCreditTokens.sub(credits);

    require(creditTokens.transferFrom(owner, msg.sender, credits) == true); //Process token purchase in th
    emit evt_TokenAction(msg.sender,credits, "Buy");
}
```

Get Player Info

```
getPlayerInfo(address _player) external view returns (uint,uint,uint8)
```

Input(s)

Player address

Return value

- Player credit token balance
- Player Ether balance
- Player isAuthorized flag

```
'  
function getPlayerInfo(address _player) external view returns (uint,uint,uint8) {  
    return (player[_player].creditBalance,player[_player].ethBalance,player[_player].isAuthorized);  
}  
  
function getCreditStatus() external view returns (uint256,uint256,uint256,uint256,uint256){  
    return (creditStatus.cap,creditStatus.authorizedPlayers,creditStatus.assignedCreditTokens,creditStatus.remainingCreditTokens,creditSta  
}  
  
function getMessageInfo(uint _messageid) external view returns (uint,address,string,bool) {  
    return (messageinfo[_messageid].CreateDate,messageinfo[_messageid].Creator,messageinfo[_messageid].MessageContent,messageinfo[_messagei  
}  
  
function kill() external onlyAdministrators { //onlyOwner can kill this contract  
    selfdestruct(owner); // `owner` is the owners address  
}
```

Get Global Credit Status

```
getCreditStatus() external view returns  
(uint256,uint256,uint256,uint256,uint256,uint256)
```

Input(s)

None

Return value

- Total credit supply (Cap)
- Total number of authorized players
- Total amount of assigned credits
- Total remaining credits
- Total amount of arised credits
- Total amount of arised Ether

Demo Script

ENVIRONAMNT DESCRIOPTION

Two smart contracts are created and deployed under Remix

CREDIT TOKEN CONTRACT (ERC20)

Check the Contract Symbol Name

```
name  
0: string: ALAKEMI
```

Check the contract Owner

```
owner  
0: address: 0xCA35b7d915458Ef540aDe6068dFe2F44E8fa733c
```

Check the contract owner token balance (By default, the owner own all 1 billion tokens)

```
balanceOf 0xca35b7d915458ef540ade6068dfe2f44e8fa733c  
0: uint256: 1000000000
```

MESSAGING CONTRACT (ALKEMIMESSAGING)

Check the “COST” as specified at the contract

This is the credit amount withdraw from user account after a message is added/removed

```
* @title AlkemiMessaging - Authorized players to add/remove messages
* @dev Implementation of AlkemiMessaging contract.
* Requirements: Create a smart contract that facilitates the storage
*   Each newly added user will have a certain credit amount that the
*   Users who are authorized should be able to purchase extra credits
*/
contract AlkemiMessaging is Pausable, ReentrancyGuard {
    using SafeMath for uint256;
    uint8 public constant COST = 1; // credit cost to add/remove a mess
    uint256 public CREDITPRICE = 3; //One Wei can buy this number of Cr
    uint256 public CREDITSONAUTHORIZED = 10; //Assigned credits take
    address public owner; // Owner of this contract
    IERC20 private creditTokens; // The Alkemi ERC Token(Credit)

    //Credit Token Global Status
    struct CreditStatus {
        uint256 cap; //Total credit token supply
        uint256 authorizedPlayers; //Total authorized players
        uint256 assignedCreditTokens; //Total assigned credit tokens or
```

removeMessage	uint256 _messageID
renouncePauser	
unpause	
withdrawCredits	uint256 _amount
COST	
0: uint8: 1	
CREDITPRICE	
0: uint256: 3	
CREDITSONAUTH	
0: uint256: 10	

Check Credit purchase base price

This is the base price to purchase credit tokens (1 Wei can buy 3 credits at this demo)

```
/*
contract AlkemiMessaging is Pausable, ReentrancyGuard {
    using SafeMath for uint256;
    uint8 public constant COST = 1; // credit cost to add/remove a mess
    uint256 public CREDITPRICE = 3; //One Wei can buy this number of Cr
    uint256 public CREDITSONAUTHORIZED = 10; //Assigned credits take
    address public owner; // Owner of this contract
    IERC20 private creditTokens; // The Alkemi ERC Token(Credit)

    //Credit Token Global Status
    struct CreditStatus {
        uint256 cap; //Total credit token supply
```

unpause	
withdrawCredits	uint256 _am
COST	
0: uint8: 1	
CREDITPRICE	
0: uint256: 3	
CREDITSONAUTH	

Check the specified credit amount assigned to new address upon they are authorized

10 credit tokens will be assigned to new address at this demo

```
/*
 * contract AlkemiMessaging is Pausable, ReentrancyGuard {
    using SafeMath for uint256;
    uint8 public constant COST = 1; // credit cost to add/remove a message
    uint256 public CREDITPRICE = 3; //One Wei can buy this number of Credits
    uint256 public CREDITSONAUTHORIZED = 10; //Assigned credits to authorized players
    address public owner; // Owner of this contract
    IERC20 private creditTokens; // The Alkemi ERC Token(Credit)

    //Credit Token Global Status
    struct CreditStatus {
        uint256 cap; //Total credit token supply
        uint256 authorizedPlayers; //Total authorized players
        uint256 assignedCreditTokens; //Total assigned credit tokens or credits
        uint256 remainingCreditTokens; //Total remaining credits (It is credits assignedCredits) //Total paid credits when player try to withdraw
    }
}
```

unpause
withdrawCredits uint256 _i
COST
0: uint8: 1
CREDITPRICE
0: uint256: 3
CREDITSONAUTHORIZED
0: uint256: 10
getCreditStatus

Switch to Owner account and deposit 10000 tokens from ERC20 credit token contract to messaging contract

```

118     require(_messageCount >= messageCount, "Only one message can be removed");
119     require(player[msg.sender].isAuthorized == 1, "This player was not
120     require(messageInfo[_messageID].isRemoved == false, "This message
121     require(player[msg.sender].creditBalance >= COST, "You don't have enough credits");
122     require(messageInfo[messageCount].Creator == msg.sender, "Only me can remove my own messages");
123     player[msg.sender].creditBalance = player[msg.sender].creditBalance - COST;
124     creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST);
125     messageInfo[messageCount].isRemoved = true; //Leave message removed
126     emit evt_MessageAction(msg.sender, _messageID, "Remove");
127     return true;
128   }
129   else {
130     /* =====Credit tokens Operation \ Deposit credit token from ERC20 */
131     function depositCredits(uint _amount) external whenNotPaused nonReentrant {
132       require(creditTokens.transferFrom(msg.sender, address(this), _amount));
133       player[msg.sender].creditBalance = player[msg.sender].creditBalance + _amount;
134       emit evt_TokenAction(msg.sender, _amount, "Deposit");
135     }
136     /* =====Credit tokens Operation \ Withdraw credit token from current account */
137     function withdrawCredits(uint _amount) external whenNotPaused nonReentrant {
138       require(player[msg.sender].creditBalance >= _amount, "You don't have enough credits");
139       require(creditTokens.transfer(msg.sender, _amount) == true);
140       player[msg.sender].creditBalance = player[msg.sender].creditBalance - _amount;
141     }
142   }
143 }
```

0: address: 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c

paused

symbol

totalSupply

(fallback)

addPauser address account

authorizeNewPlayer address _newplayer

buyCredits

createMessage string _messagecontent

depositCredits 10000

kill

pause

removeMessage uint256 _messageID

renouncePauser

unpause

withdrawCredits 1000000000

COST

0: uint8: 1

CREDITPRICE

0: uint256: 3

CREDITSOURCEAUTHORIZED

0: uint256: 10

getCreditStatus

0: uint256: 1000000000

1: uint256: 0

Specified owner address and check 1000 tokens are deposited to owner account successfully via following function at the messaging contract

getPlayerInfo	0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c
0: uint256: 10000	
1: uint256: 0	
2: uint8: 0	

Authorize a new address by owner

- ◆ Get another account address

```
//, -----  
valid Message ID");  
d == 1,"This player was not  
ved == false,"This message  
ce >= COST , "You don't hav  
tor == msg.sender , "Only me  
yer[msg.sender].creditBalanc  
tus.arisedCredits.add(COST)  
true; //Leave message rem  
ssageID, "Remove");  
  
sit credit token from ERC20  
ernal whenNotPaused nonReen  
msg.sender, address(this),  
player[msg.sender].creditB  
amount, "Deposit").  
//, -----
```

Environment JavaScript VM VM (-) i

Account 0xca3...a733c (99.9999999995492701)

Gas limit 0xca3...a733c (99.99999999954927019 ether)

Value 0x147...c160c (99.99999999995850659 ether)
0x4b0...4d2db (100 ether)
0x583...40225 (100 ether)
0xdd8...92148 (100 ether)

AlkemiMessaging

Deploy 0x2b47f0d926a28adfc90a69939502dcb687ac3686

or

At Address Load contract from Address

- ◆ Before the authorization, Get the new address credit balance.

Paste the address and get the credit balance before the authorization (it is 0 at this demo)

getPlayerInfo 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

0: uint256: 0

1: uint256: 0

2: uint8: 0

- ◆ Execute the new address authorization

Make sure switch to owner account, Specify the new address and execute the new address authorization

```

74
75 //Authorized a new player (Dener only / Not allow Reentrancy / block
76 function authorizeNewPlayer(address _newplayer) external onlyAdminis
77   require(_newplayer != address(0),"Please specify new player ac
78   require(player[_newplayer].isAuthorized == 0,"This player alre
79   require(creditStatus.remainingCreditTokens > CREDITSONAUTHORIZ
80   require(player[owner].creditBalance >= CREDITSONAUTHORIZED , "
81
82   //Transfer credit tokens from owner account to new player acc
83   player[owner].creditBalance = player[owner].creditBalance.sut
84   player[_newplayer].creditBalance = player[_newplayer].creditB
85   //Update global credit status
86   creditStatus.remainingCreditTokens = creditStatus.remainingCr
87   creditStatus.assignedCreditTokens = creditStatus.assignedCred
88   creditStatus.authorizedPlayers++;
89
90   player[_newplayer].isAuthorized = 1; //Assign 1 to authorized
91   emit evt_AuthorizedNewPlayer(msg.sender,_newplayer);
92 }
93 }
```

[2] only remix transactions, script ▾ Search transaction

[vm]	from:0xca3...a733c
	to:AlkemiMessaging.authorizeNewPlayer(address) 0xbca...1
	value: wei data:0x4e7...c160c logs:1 hash:0xcb1...93c69
status	0x1 Transaction mined and execution succeeded
transaction hash	0xcb15123f98a0dde3b7e3ba2c5e7c1c659a66b65182f08ef522
from	0xca35b7d915458ef540ade6068dfc2f44e8fa733c
to	AlkemiMessaging.authorizeNewPlayer(address) 0xbcaafb
gas	5000000 gas
transaction cost	126213 gas
execution cost	103533 gas
hash	0xcb15123f98a0dde3b7e3ba2c5e7c1c659a66b65182f08ef522
input	0x4e7...c160c
decoded input	{ "address _newplayer": "0x14723A09ACFF6D2A600 cdf7a4AFF308FDDC160C" }

AlkemiMessaging at 0xbca...193ef (memory)	
(fallback)	
addPauser	address account
authorizeNewPlaye	0x14723a09acff6d2a600dcdf7aa4aff308fddc160c
buyCredits	
createMessage	string _messagecontent
depositCredits	10000
kill	
pause	
removeMessage	uint256 _messageID
renouncePauser	
unpause	
withdrawCredits	1000000000
COST	
0: uint8: 1	
CREDITPRICE	
0: uint256: 3	
CREDITSONAUTHORIZED	
0: uint256: 10	
getCreditStatus	
0: uint256: 1000000000	
1: uint256: 0	
2: uint256: 0	
3: uint256: 1000000000	

- ◆ Verify 10 tokens are assigned to the new address and the authorization flag is set as “1”

getPlayerInfo	0x14723a09acff6d2a600dcdf7aa4aff308fddc160c
0: uint256: 10	
1: uint256: 0	
2: uint8: 1	

isPauser address account

- ◆ Verify the global credit status is updated Such as, at following sample.
 - The total assigned player” is set as 1.
 - total assigned credits is set as “10”
 - 10 was reduced from the total remaining credits

getCreditStatus

```

0: uint256: 10000000000
1: uint256: 1
2: uint256: 10
3: uint256: 999999990
4: uint256: 0
5: uint256: 0

```

```

; 68 creditStatus.cap = token.totalSupply(); //Total credit token
69 creditStatus.assignedCreditTokens = 0; //the number of credit
70 creditStatus.remainingCreditTokens = creditStatus.cap; //num
71 creditStatus.arisedEth = 0; // total arised Ether in Wei
72 creditTokens = token; //Credit token contract address
73 }

//Authorized a new player (Owner only / Not allow Reentrancy / block
74
75 function authorizeNewPlayer(address _newplayer) external onlyAdminis
76 require(_newplayer != address(0), "Please specify new player account");
77 require(player[_newplayer].isAuthorized == 0, "This player already authorized");
78 require(creditStatus.remainingCreditTokens > CREDITSONAUTHORIZERED);
79 require(player[owner].creditBalance >= CREDITSONAUTHORIZED, "Insufficient balance");
80
81 //Transfer credit tokens from owner account to new player account
82 player[owner].creditBalance = player[owner].creditBalance.sub(
83 player[_newplayer].creditBalance = player[_newplayer].creditBalance.add(
84 //Update global credit status
85 creditStatus.remainingCreditTokens = creditStatus.remainingCreditTokens - CREDITSONAUTHORIZERED;
86 creditStatus.assignedCreditTokens = creditStatus.assignedCreditTokens + CREDITSONAUTHORIZERED;
87 creditStatus.authorizedPlayers++;
88
89
90 player[_newplayer].isAuthorized = 1; //Assign 1 to authorized
91 emit evt_AuthorizedNewPlayer(msg.sender,_newplayer);
92
93 }
94

```

[2] only remix transactions, script ▾ Search transaction

transact to AlkemiMessaging.authorizeNewPlayer pending ...

[vm] from:0xca3...a733c to:AlkemiMessaging.authorizeNewPlayer(address) 0xbca...193e Debug ▾ value:0 wei data:0x4e7...c160c logs:0 hash:0x99e...49259

transact to AlkemiMessaging.authorizeNewPlayer errored: VM error: revert. revert_ The transaction has been reverted to the initial state. Reason provided by the contract: "This player already was authorized". Debug the transaction to get more information.

AlkemiMessaging at 0xbca...193ef (menu)

(fallback)	
addPauser	address account
authorizeNewPlayer	0x14723a09acff8d2a80dcdf7aa4aff308fddc160c
buyCredits	
createMessage	string _messagecontent
depositCredits	10000
kill	
pause	
removeMessage	uint256 _messageID
renouncePauser	
unpause	
withdrawCredits	1000000000
COST	
0: uint8: 1	
CREDITPRICE	
0: uint256: 3	
CREDITSONAUTHORIZERED	
0: uint256: 10	
getCreditStatus	
0: uint256: 10000000000	
1: uint256: 1	
2: uint256: 10	
3: uint256: 999999990	
4: uint256: 0	

- ◆ Execute the new address authorization one more time with the same address. The authorization will be declined because that address already was authorized.

```

68     creditStatus.cap = token.totalSupply(); //Total credit take
69     creditStatus.assignedCreditTokens = 0; //the number of credit
70     creditStatus.remainingCreditTokens = creditStatus.cap; //num
71     creditStatus.arisedEth = 0; // total arised Ether in Wei
72     creditTokens = token; //Credit token contract address
73 }
74
75 //Authorized a new player (Owner only / Not allow Reentrancy / block
76 function authorizeNewPlayer(address _newplayer) external onlyAdminis-
77   require(_newplayer != address(0), "Please specify new player ac-
78   require(player[_newplayer].isAuthorized == 0, "This player alre-
79   require(creditStatus.remainingCreditTokens > CREDITSONAUTHORIZED);
80   require(player[owner].creditBalance >= CREDITSONAUTHORIZED, "C-
81
82   //Transfer credit tokens from owner account to new player acc
83   player[owner].creditBalance = player[owner].creditBalance.su-
84   player[_newplayer].creditBalance = player[_newplayer].credi-
85   //Update global credit status
86   creditStatus.remainingCreditTokens = creditStatus.remainingCr-
87   creditStatus.assignedCreditTokens = creditStatus.assignedCre-
88   creditStatus.authorizedPlayers++;
89
90   player[_newplayer].isAuthorized = 1; //Assign 1 to authoriz-
91
92   emit evt_AuthorizedNewPlayer(msg.sender,_newplayer);
93 }
94

```

transact to AlkemiMessaging.authorizeNewPlayer pending ...

[vm] from:0xca3...a733c to:AlkemiMessaging.authorizeNewPlayer(address) 0xbca...193e Debug value:0 wei data:0x4e7...c160c logs:0 hash:0x99e..49259

transact to AlkemiMessaging.authorizeNewPlayer errored: VM error: revert. revert The transaction has been reverted to the initial state. Reason provided by the contract: "This player already was authorized". Debug the transaction to get more information.

AlkemiMessaging at 0xbca...193ef (men)	
(fallback)	
addPauser	address account
authorizeNewPlaye	0x14723a09acf0d2a80ddcf7aa4aff308fddc160c
buyCredits	
createMessage	string _messagecontent
depositCredits	10000
kill	
pause	
removeMessage	uint256 _messageID
renouncePauser	
unpause	
withdrawCredits	1000000000
COST	
0: uint8: 1	
CREDITPRICE	
0: uint256: 3	
CREDITSONAUTHORIZED	
0: uint256: 10	
getCreditStatus	
0: uint256: 1000000000	
1: uint256: 1	
2: uint256: 10	
3: uint256: 999999990	
4: uint256: 0	

Add a new message / Remove a message

- ◆ Switch to above authorized address

Environment: JavaScript VM
 Account: 0xca3...a733c (99.9999999995492701)
 Gas limit: 0x147...c160c (99.999999999995850659 ether)
 Value: 0xb0...4d2db (100 ether)

- ◆ Specify any message content and execute "createmessage" function
 The function is executed successfully and returns the message ID "1" at following sample

```

90 FUNCTION createmessage(string _messagecontent) EXTERNAL WHENNOTPAUSED ISHUMAN NOT
91 require(player[msg.sender].isAuthorized == 1, "This player was not authorized")
92 require(player[msg.sender].creditBalance >= COST, "You don't have enough credit")
93
94     messageCount++;
95
96     player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(COST);
97
98     creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST); //Update credit
99
100
101     //Record added message info
102     messageInfo[messageCount].CreateDate = now;
103     messageInfo[messageCount].Creator = msg.sender;
104     messageInfo[messageCount].MessageContent = _messagecontent;
105
106     emit evt_MessageAction(msg.sender, messageCount, "Add");
107     return messageCount; //return added message ID
108 }
109
110
111 /* =====Messaging facilitates \ Remove an existing Message(Not allow Reentrancy)
112 function removeMessage(uint256 _messageID) EXTERNAL WHENNOTPAUSED ISHUMAN NOT
113 require(_messageID <= messageCount, "Invalid Message ID");
114 require(player[msg.sender].isAuthorized == 1, "This player was not authorized")
115 require(messageInfo[_messageID].isRemoved == false, "This message already removed")
116 require(player[msg.sender].creditBalance >= COST, "You don't have enough credits")
117 require(messageInfo[_messageCount].Creator == msg.sender, "Only message creator can remove")
118
119     player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(COST);
120
121
122
123
124
  
```

[2] only remix transactions, script ▾ Search transactions

to:AlkemiMessaging.createmessage(string) 0xbca...193ef	value:0 Wei
data:0x198...00000 logs:1 hash:0x19d...528ce	
status	0x1 Transaction mined and execution succeed
transaction hash	0x19d8a277d5e7b85976c88d263c3ffa10d343824d2358cc3e40a68ecf2e4528ce
from	AlkemiMessaging.createMessage(string) 0xbcaafbf4802c27917d29449c8ab707ff6f86193ef
gas	5000000 gas
transaction cost	181965 gas
execution cost	157685 gas
gas	0x19d8a277d5e7b85976c88d263c3ffa10d343824d2358cc3e40a68ecf2e4528ce
input	0x198...00000
encoded input	{ "string _messagecontent": "Message created by authorized address" }
encoded output	{ "0": "uint256: 1" }

AlkemiMessaging at 0xbca...193ef (me)
(fallback)
addPauser address account
authorizeNewPlayer 0x14723a09acf6d2a60dcdf7aa4aff308fddc16c
buyCredits
createMessage "Message created by authorized address"
depositCredits 10000
kill
pause
removeMessage uint256 _messageID
renouncePauser
unpause
withdrawCredits 100000000
cost
0: uint8: 1
CREDITPRICE
0: uint256: 3
CREDITSAUTHORIZED
0: uint256: 10
getCreditStatus
0: uint256: 1000000000
1: uint256: 1
2: uint256: 10
3: uint256: 999999990
4: uint256: 0
5: uint256: 0
getMessageInfo uint256 _messageid
getPlayerInfo 0x14723a09acf6d2a60dcdf7aa4aff308fddc16c

- ◆ Specify the message ID “1” and execute “getMessageInfo”
It returns the message details (such as message creation date/time, message creator address, and message content and isremoved flag)

The screenshot shows the Remix IDE interface with the Alkemi Messaging contract deployed. The left side displays the Solidity code for the messaging logic, including functions like `sendMessage`, `removeMessage`, and `getMessageInfo`. The right side shows the transaction history and pending transaction details.

Pending Transaction:

```
[vm] from:0xca3...a733c
to:AlkemiMessaging.removeMessage(uint256) 0xbca...193ef value:0 wei
data:0x2de...00001 logs:0 hash:0x7b4...3455
```

Transaction History:

Index	Function	Value
0	createMessage	"Message created by authorized address"
1	depositCredits	10000
2	kill	
3	pause	
4	removeMessage	1
5	renouncePauser	
6	unpause	
7	withdrawCredits	1000000000
8	COST	
9	0: uint8: 1	
10	CREDITPRICE	
11	0: uint256: 3	
12	CREDITSONAUTHORIZED	
13	0: uint256: 10	
14	getCreditStatus	
15	0: uint256: 1000000000	
16	1: uint256: 1	
17	2: uint256: 10	
18	3: uint256: 999999990	
19	4: uint256: 0	
20	5: uint256: 0	
21	getMessageInfo	1
22	0: uint256: 1569943808	
23	1: address: 0x14723A09ACff6D2A60DcdF7aA4AFF308FDDC160C	
24	2: string: Message created by authorized address	
25	3: bool: false	

- ◆ Stay at the same address and execute the “removemessage” with the messageId as “1”

The message flag “isRemoved” is changed from “false” to “true” (see the second screen below)

One more credits are withdraw from current player account (When the address is authorized, there are 10 credits are assigned at his account. It reduces to 8 after adding and removing a message. Please check enclosed the Third screen for detail)

```

100     messageCount++;
101
102     player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(COST);
103
104     creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST); //Update g
105
106     //Record added message info
107     messageinfo[messageCount].CreateDate = now;
108     messageinfo[messageCount].Creator = msg.sender;
109     messageinfo[messageCount].MessageContent = _messagecontent;
110
111     emit evt_MessageAction(msg.sender, messageCount, "Add");
112     return messageCount; //return added message ID
113 }
114
115 /* ======Messaging facilitates \ Remove an existing Message(Not allow Reentranc
116 function removeMessage(uint256 _messageID) external whenNotPaused isHuman nonRe
117 require(_messageID <= messageCount); "Invalid Message ID");
118 require(player[msg.sender].isAuthorized == 1, "This player was not authorized y
119 require(messageinfo[_messageID].isRemoved == false, "This message already remov
120 require(player[msg.sender].creditBalance > COST, "You don't have no enough c
121 require(messageinfo[messageCount].Creator == msg.sender, "Only message creator
122
123     player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(COST); ▶
124

```

[2] only remix transactions, script ▾

status	0x1 Transaction mined and execution succeed
transaction hash	0xa6009b1faa72b104480eaef60eae40a40c139ea216a192f0e3299cd5890c3ad4
from	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
to	AlkemiMessaging.removeMessage(uint256) 0xbcaafb4802c27917d29449c8ab70ff6f86193ef
gas	5000000 gas
transaction cost	63370 gas
execution cost	41906 gas
hash	0xa6009b1faa72b104480eaef60eae40a40c139ea216a192f0e3299cd5890c3ad4
input	0x2de...00001
decoded input	{ "uint256 _messageID": "1" }
decoded output	{ "0": "bool: true" }
logs	[]

getMessageInfo 1

0: uint256: 1569943808
1: address: 0x14723A09ACff6D2A60DcdF7aA4AFF308FDDC160C
2: string: Message created by authorized address
3: bool: true

getPlayerInfo

0: uint256: 8
1: uint256: 0
2: uint8: 1

Buy Credits

- ◆ Switch to any authorized address
- ◆ Get the credit balance at the ERC20 token contract before the buying. Such as: the balance is 0 for that address at the ERC20 contract

Deployed Contracts

The screenshot shows the Alkemi messaging contract interface. At the top, it displays the contract name "ALAKEMI at 0x2b4...c3686 (memory)". Below this, a list of functions is shown with their parameters:

- addPauser**: address account
- approve**: 0xbcaaf84802c27917d29449c8ab707ff8f86193ef,100
- pause**
- renouncePauser**
- transfer**: address to, uint256 value
- transferFrom**: address from, address to, uint256 value
- unpause**
- allowance**: address _owner, address spender
- balanceOf**: 0x14723a09acf8d2a60ddcf7aa4aff308fddc160c

Below the functions, the value "0: uint256: 0" is displayed. Further down, there are two more function entries:

- isPauser**: address account
- name**

- ◆ Verify the credit price before the buying. It is set as "1 Wei can buy 3 credits" at following sample. Such as if current address paid 4 Wei, there should be 12 credits deposited to his balance

The screenshot shows the source code for the Alkemi Messaging contract and its deployment parameters.

Source Code (Lines 1-20):

```

1 pragma solidity ^0.4.24;
2
3 import "./IERC20.sol";
4 import "./SafeMath.sol";
5 import "./Pausable.sol";
6 import "./ReentrancyGuard.sol";
7
8 /**
9  * @title AlkemiMessaging - Authorized players to add/remove messages
10 * @dev Implementation of AlkemiMessaging contract.
11 * Requirements: Create a smart contract that facilitates the storage of messages on-chain. The s
12 *   Each newly added user will have a certain credit amount that they are assigned on add. From
13 *   Users who are authorized should be able to purchase extra credits by sending funds to the sm
14 */
15 contract AlkemiMessaging is Pausable, ReentrancyGuard {
16     using SafeMath for uint256;
17     uint8 public constant COST = 1; // credit cost to add/remove a message
18     uint256 public CREDITPRICE = 3; //One Wei can buy this number of Credit tokens
19     uint256 public CREDITSONAUTHORIZED = 10; //Assigned credits tokens on authorized by the owne
20 }
```

Deployment Parameters:

- withdrawCredits**: 1000000000
- COST**: 0: uint8: 1
- CREDITPRICE**: 0: uint256: 3
- CREDITSONAUTHORIZED**: 0: uint256: 10
- getCreditStatus**: 0: uint256: 1000000000

- ◆ Verify the global credit status. After the buying, the ether amount should be added to the arisedEther (Before the buying, the arisedEther is 0 at following sample)

The screenshot shows the Alkemi Messaging contract in the Remix IDE. The code includes a withdrawCredits function with a value of 1000000000. The interface displays various parameters and their values, such as COST (0), CREDITPRICE (0), and CREDITSONAUTHORIZED (0).

```

4 //Credit Token Global Status
5 struct CreditStatus {
6     uint256 cap; //Total credit token supply
7     uint256 authorizedPlayers; //Total authorized players
8     uint256 assignedCreditTokens; //Total assigned credit tokens on authorized new player
9     uint256 remainingCreditTokens; //Total remaining credits (It is total supply credits by def
10    uint256 arisedCredits; //Total paid credits when player try to add/remove messages
11    uint256 arisedEth; //Total Arised Ether in Wei (Ether collected when player buy credit
12 }
13 CreditStatus creditStatus;
14
15 //Message Info
16 struct MessageInfo {
17     uint CreateDate; //Message creation date/time in UNIX datetime format
18     address Creator; //Message owner(creator). only message creator can remove this message
19 }
20

```

◆ Specify the value as “4” Wei and execute the “buyCredits” function

The screenshot shows the Alkemi Messaging contract in the Remix IDE. The code includes a buyCredits function with a value of 4 wei. The interface displays the transaction details, including the status, transaction hash, from, to, gas, transaction cost, execution cost, hash, input, decoded input, decoded output, and logs.

```

9 * @title AlkemiMessaging - AutnORIZED players to add/remove messages
10 * @dev Implementation of AlkemiMessaging contract.
11 * Requirements: Create a smart contract that facilitates the storage of messages on-chain. The s
12 *   * Each newly added user will have a certain credit amount that they are assigned on add. From
13 *   * Users who are authorized should be able to purchase extra credits by sending funds to the sm
14 */
15 contract AlkemiMessaging is Pausable, ReentrancyGuard {
16     using SafeMath for uint256;
17     uint public constant COST = 1; // credit cost to add/remove a message
18     uint256 public CREDITPRICE = 3; //One Wei can buy this number of Credit tokens
19     uint256 public CREDITSONAUTHORIZED = 10; //Assigned credits tokens on authorized by the owne
20
21     address public owner; // Owner of this contract
22
23     IERC20 private creditTokens; // The Alkemi ERC Token(Credit)
24
25     //Credit Token Global Status
26     struct CreditStatus {
27         uint256 cap; //Total credit token supply
28         uint256 authorizedPlayers; //Total authorized players
29         uint256 assignedCreditTokens; //Total assigned credit tokens on authorized new player
30         uint256 remainingCreditTokens; //Total remaining credits (It is total supply credits by def
31         uint256 arisedCredits; //Total paid credits when player try to add/remove messages
32         uint256 arisedEth; //Total Arised Ether in Wei (Ether collected when player buy credit
33     }
34     CreditStatus creditStatus;
35
36     //Message Info
37     struct MessageInfo {
38         uint CreateDate; //Message creation date/time in UNIX datetime format
39         address Creator; //Message owner(creator). only message creator can remove this message
40

```

status	0x1 Transaction mined and execution succeed
transaction hash	0x1205a79871266e4aaef38ef1c6aba4aac0b4147b395bfae40f2ade818c11e42e
from	0x14723a0acfdd2a660dcdf7aa4aff308ffddc180c
to	AlkemiMessaging.buyCredits()
gas	5000000 gas
transaction cost	102167 gas
execution cost	80895 gas
hash	0x912b5a79871266e4aaef38ef1c6aba4aac0b4147b395bfae40f2ade818c11e42e
input	0x83f...31cd3
decoded input	{}
decoded output	{}
logs	[{"from": "0x2b47f0d926a28adfc90a69939502dc687ac3686", "topic": "0xd0f252ad1be2c80b69c2b068fc378da952ba7f163ca11628f55a4df523bief", "event": "Transfer", "args": {"arg1": "0x2b47f0d926a28adfc90a69939502dc687ac3686", "arg2": "0x0", "arg3": "0x0", "arg4": "0x0", "arg5": "0x0", "arg6": "0x0", "arg7": "0x0", "arg8": "0x0", "arg9": "0x0", "arg10": "0x0", "arg11": "0x0", "arg12": "0x0", "arg13": "0x0", "arg14": "0x0", "arg15": "0x0", "arg16": "0x0", "arg17": "0x0", "arg18": "0x0", "arg19": "0x0", "arg20": "0x0", "arg21": "0x0", "arg22": "0x0", "arg23": "0x0", "arg24": "0x0", "arg25": "0x0", "arg26": "0x0", "arg27": "0x0", "arg28": "0x0", "arg29": "0x0", "arg30": "0x0", "arg31": "0x0", "arg32": "0x0", "arg33": "0x0", "arg34": "0x0", "arg35": "0x0", "arg36": "0x0", "arg37": "0x0", "arg38": "0x0", "arg39": "0x0", "arg40": "0x0", "arg41": "0x0", "arg42": "0x0", "arg43": "0x0", "arg44": "0x0", "arg45": "0x0", "arg46": "0x0", "arg47": "0x0", "arg48": "0x0", "arg49": "0x0", "arg50": "0x0", "arg51": "0x0", "arg52": "0x0", "arg53": "0x0", "arg54": "0x0", "arg55": "0x0", "arg56": "0x0", "arg57": "0x0", "arg58": "0x0", "arg59": "0x0", "arg60": "0x0", "arg61": "0x0", "arg62": "0x0", "arg63": "0x0", "arg64": "0x0", "arg65": "0x0", "arg66": "0x0", "arg67": "0x0", "arg68": "0x0", "arg69": "0x0", "arg70": "0x0", "arg71": "0x0", "arg72": "0x0", "arg73": "0x0", "arg74": "0x0", "arg75": "0x0", "arg76": "0x0", "arg77": "0x0", "arg78": "0x0", "arg79": "0x0", "arg80": "0x0", "arg81": "0x0", "arg82": "0x0", "arg83": "0x0", "arg84": "0x0", "arg85": "0x0", "arg86": "0x0", "arg87": "0x0", "arg88": "0x0", "arg89": "0x0", "arg90": "0x0", "arg91": "0x0", "arg92": "0x0", "arg93": "0x0", "arg94": "0x0", "arg95": "0x0", "arg96": "0x0", "arg97": "0x0", "arg98": "0x0", "arg99": "0x0", "arg100": "0x0", "arg101": "0x0", "arg102": "0x0", "arg103": "0x0", "arg104": "0x0", "arg105": "0x0", "arg106": "0x0", "arg107": "0x0", "arg108": "0x0", "arg109": "0x0", "arg110": "0x0", "arg111": "0x0", "arg112": "0x0", "arg113": "0x0", "arg114": "0x0", "arg115": "0x0", "arg116": "0x0", "arg117": "0x0", "arg118": "0x0", "arg119": "0x0", "arg120": "0x0", "arg121": "0x0", "arg122": "0x0", "arg123": "0x0", "arg124": "0x0", "arg125": "0x0", "arg126": "0x0", "arg127": "0x0", "arg128": "0x0", "arg129": "0x0", "arg130": "0x0", "arg131": "0x0", "arg132": "0x0", "arg133": "0x0", "arg134": "0x0", "arg135": "0x0", "arg136": "0x0", "arg137": "0x0", "arg138": "0x0", "arg139": "0x0", "arg140": "0x0", "arg141": "0x0", "arg142": "0x0", "arg143": "0x0", "arg144": "0x0", "arg145": "0x0", "arg146": "0x0", "arg147": "0x0", "arg148": "0x0", "arg149": "0x0", "arg150": "0x0", "arg151": "0x0", "arg152": "0x0", "arg153": "0x0", "arg154": "0x0", "arg155": "0x0", "arg156": "0x0", "arg157": "0x0", "arg158": "0x0", "arg159": "0x0", "arg160": "0x0", "arg161": "0x0", "arg162": "0x0", "arg163": "0x0", "arg164": "0x0", "arg165": "0x0", "arg166": "0x0", "arg167": "0x0", "arg168": "0x0", "arg169": "0x0", "arg170": "0x0", "arg171": "0x0", "arg172": "0x0", "arg173": "0x0", "arg174": "0x0", "arg175": "0x0", "arg176": "0x0", "arg177": "0x0", "arg178": "0x0", "arg179": "0x0", "arg180": "0x0", "arg181": "0x0", "arg182": "0x0", "arg183": "0x0", "arg184": "0x0", "arg185": "0x0", "arg186": "0x0", "arg187": "0x0", "arg188": "0x0", "arg189": "0x0", "arg190": "0x0", "arg191": "0x0", "arg192": "0x0", "arg193": "0x0", "arg194": "0x0", "arg195": "0x0", "arg196": "0x0", "arg197": "0x0", "arg198": "0x0", "arg199": "0x0", "arg200": "0x0", "arg201": "0x0", "arg202": "0x0", "arg203": "0x0", "arg204": "0x0", "arg205": "0x0", "arg206": "0x0", "arg207": "0x0", "arg208": "0x0", "arg209": "0x0", "arg210": "0x0", "arg211": "0x0", "arg212": "0x0", "arg213": "0x0", "arg214": "0x0", "arg215": "0x0", "arg216": "0x0", "arg217": "0x0", "arg218": "0x0", "arg219": "0x0", "arg220": "0x0", "arg221": "0x0", "arg222": "0x0", "arg223": "0x0", "arg224": "0x0", "arg225": "0x0", "arg226": "0x0", "arg227": "0x0", "arg228": "0x0", "arg229": "0x0", "arg230": "0x0", "arg231": "0x0", "arg232": "0x0", "arg233": "0x0", "arg234": "0x0", "arg235": "0x0", "arg236": "0x0", "arg237": "0x0", "arg238": "0x0", "arg239": "0x0", "arg240": "0x0", "arg241": "0x0", "arg242": "0x0", "arg243": "0x0", "arg244": "0x0", "arg245": "0x0", "arg246": "0x0", "arg247": "0x0", "arg248": "0x0", "arg249": "0x0", "arg250": "0x0", "arg251": "0x0", "arg252": "0x0", "arg253": "0x0", "arg254": "0x0", "arg255": "0x0", "arg256": "0x0", "arg257": "0x0", "arg258": "0x0", "arg259": "0x0", "arg260": "0x0", "arg261": "0x0", "arg262": "0x0", "arg263": "0x0", "arg264": "0x0", "arg265": "0x0", "arg266": "0x0", "arg267": "0x0", "arg268": "0x0", "arg269": "0x0", "arg270": "0x0", "arg271": "0x0", "arg272": "0x0", "arg273": "0x0", "arg274": "0x0", "arg275": "0x0", "arg276": "0x0", "arg277": "0x0", "arg278": "0x0", "arg279": "0x0", "arg280": "0x0", "arg281": "0x0", "arg282": "0x0", "arg283": "0x0", "arg284": "0x0", "arg285": "0x0", "arg286": "0x0", "arg287": "0x0", "arg288": "0x0", "arg289": "0x0", "arg290": "0x0", "arg291": "0x0", "arg292": "0x0", "arg293": "0x0", "arg294": "0x0", "arg295": "0x0", "arg296": "0x0", "arg297": "0x0", "arg298": "0x0", "arg299": "0x0", "arg300": "0x0", "arg301": "0x0", "arg302": "0x0", "arg303": "0x0", "arg304": "0x0", "arg305": "0x0", "arg306": "0x0", "arg307": "0x0", "arg308": "0x0", "arg309": "0x0", "arg310": "0x0", "arg311": "0x0", "arg312": "0x0", "arg313": "0x0", "arg314": "0x0", "arg315": "0x0", "arg316": "0x0", "arg317": "0x0", "arg318": "0x0", "arg319": "0x0", "arg320": "0x0", "arg321": "0x0", "arg322": "0x0", "arg323": "0x0", "arg324": "0x0", "arg325": "0x0", "arg326": "0x0", "arg327": "0x0", "arg328": "0x0", "arg329": "0x0", "arg330": "0x0", "arg331": "0x0", "arg332": "0x0", "arg333": "0x0", "arg334": "0x0", "arg335": "0x0", "arg336": "0x0", "arg337": "0x0", "arg338": "0x0", "arg339": "0x0", "arg340": "0x0", "arg341": "0x0", "arg342": "0x0", "arg343": "0x0", "arg344": "0x0", "arg345": "0x0", "arg346": "0x0", "arg347": "0x0", "arg348": "0x0", "arg349": "0x0", "arg350": "0x0", "arg351": "0x0", "arg352": "0x0", "arg353": "0x0", "arg354": "0x0", "arg355": "0x0", "arg356": "0x0", "arg357": "0x0", "arg358": "0x0", "arg359": "0x0", "arg360": "0x0", "arg361": "0x0", "arg362": "0x0", "arg363": "0x0", "arg364": "0x0", "arg365": "0x0", "arg366": "0x0", "arg367": "0x0", "arg368": "0x0", "arg369": "0x0", "arg370": "0x0", "arg371": "0x0", "arg372": "0x0", "arg373": "0x0", "arg374": "0x0", "arg375": "0x0", "arg376": "0x0", "arg377": "0x0", "arg378": "0x0", "arg379": "0x0", "arg380": "0x0", "arg381": "0x0", "arg382": "0x0", "arg383": "0x0", "arg384": "0x0", "arg385": "0x0", "arg386": "0x0", "arg387": "0x0", "arg388": "0x0", "arg389": "0x0", "arg390": "0x0", "arg391": "0x0", "arg392": "0x0", "arg393": "0x0", "arg394": "0x0", "arg395": "0x0", "arg396": "0x0", "arg397": "0x0", "arg398": "0x0", "arg399": "0x0", "arg400": "0x0", "arg401": "0x0", "arg402": "0x0", "arg403": "0x0", "arg404": "0x0", "arg405": "0x0", "arg406": "0x0", "arg407": "0x0", "arg408": "0x0", "arg409": "0x0", "arg410": "0x0", "arg411": "0x0", "arg412": "0x0", "arg413": "0x0", "arg414": "0x0", "arg415": "0x0", "arg416": "0x0", "arg417": "0x0", "arg418": "0x0", "arg419": "0x0", "arg420": "0x0", "arg421": "0x0", "arg422": "0x0", "arg423": "0x0", "arg424": "0x0", "arg425": "0x0", "arg426": "0x0", "arg427": "0x0", "arg428": "0x0", "arg429": "0x0", "arg430": "0x0", "arg431": "0x0", "arg432": "0x0", "arg433": "0x0", "arg434": "0x0", "arg435": "0x0", "arg436": "0x0", "arg437": "0x0", "arg438": "0x0", "arg439": "0x0", "arg440": "0x0", "arg441": "0x0", "arg442": "0x0", "arg443": "0x0", "arg444": "0x0", "arg445": "0x0", "arg446": "0x0", "arg447": "0x0", "arg448": "0x0", "arg449": "0x0", "arg450": "0x0", "arg451": "0x0", "arg452": "0x0", "arg453": "0x0", "arg454": "0x0", "arg455": "0x0", "arg456": "0x0", "arg457": "0x0", "arg458": "0x0", "arg459": "0x0", "arg460": "0x0", "arg461": "0x0", "arg462": "0x0", "arg463": "0x0", "arg464": "0x0", "arg465": "0x0", "arg466": "0x0", "arg467": "0x0", "arg468": "0x0", "arg469": "0x0", "arg470": "0x0", "arg471": "0x0", "arg472": "0x0", "arg473": "0x0", "arg474": "0x0", "arg475": "0x0", "arg476": "0x0", "arg477": "0x0", "arg478": "0x0", "arg479": "0x0", "arg480": "0x0", "arg481": "0x0", "arg482": "0x0", "arg483": "0x0", "arg484": "0x0", "arg485": "0x0", "arg486": "0x0", "arg487": "0x0", "arg488": "0x0", "arg489": "0x0", "arg490": "0x0", "arg491": "0x0", "arg492": "0x0", "arg493": "0x0", "arg494": "0x0", "arg495": "0x0", "arg496": "0x0", "arg497": "0x0", "arg498": "0x0", "arg499": "0x0", "arg500": "0x0", "arg501": "0x0", "arg502": "0x0", "arg503": "0x0", "arg504": "0x0", "arg505": "0x0", "arg506": "0x0", "arg507": "0x0", "arg508": "0x0", "arg509": "0x0", "arg510": "0x0", "arg511": "0x0", "arg512": "0x0", "arg513": "0x0", "arg514": "0x0", "arg515": "0x0", "arg516": "0x0", "arg517": "0x0", "arg518": "0x0", "arg519": "0x0", "arg520": "0x0", "arg521": "0x0", "arg522": "0x0", "arg523": "0x0", "arg524": "0x0", "arg525": "0x0", "arg526": "0x0", "arg527": "0x0", "arg528": "0x0", "arg529": "0x0", "arg530": "0x0", "arg531": "0x0", "arg532": "0x0", "arg533": "0x0", "arg534": "0x0", "arg535": "0x0", "arg536": "0x0", "arg537": "0x0", "arg538": "0x0", "arg539": "0x0", "arg540": "0x0", "arg541": "0x0", "arg542": "0x0", "arg543": "0x0", "arg544": "0x0", "arg545": "0x0", "arg546": "0x0", "arg547": "0x0", "arg548": "0x0", "arg549": "0x0", "arg550": "0x0", "arg551": "0x0", "arg552": "0x0", "arg553": "0x0", "arg554": "0x0", "arg555": "0x0", "arg556": "0x0", "arg557": "0x0", "arg558": "0x0", "arg559": "0x0", "arg560": "0x0", "arg561": "0x0", "arg562": "0x0", "arg563": "0x0", "arg564": "0x0", "arg565": "0x0", "arg566": "0x0", "arg567": "0x0", "arg568": "0x0", "arg569": "0x0", "arg570": "0x0", "arg571": "0x0", "arg572": "0x0", "arg573": "0x0", "arg574": "0x0", "arg575": "0x0", "arg576": "0x0", "arg577": "0x0", "arg578": "0x0", "arg579": "0x0", "arg580": "0x0", "arg581": "0x0", "arg582": "0x0", "arg583": "0x0", "arg584": "0x0", "arg585": "0x0", "arg586": "0x0", "arg587": "0x0", "arg588": "0x0", "arg589": "0x0", "arg590": "0x0", "arg591": "0x0", "arg592": "0x0", "arg593": "0x0", "arg594": "0x0", "arg595": "0x0", "arg596": "0x0", "arg597": "0x0", "arg598": "0x0", "arg599": "0x0", "arg600": "0x0", "arg601": "0x0", "arg602": "0x0", "arg603": "0x0", "arg604": "0x0", "arg605": "0x0", "arg606": "0x0", "arg607": "0x0", "arg608": "0x0", "arg609": "0x0", "arg610": "0x0", "arg611": "0x0", "arg612": "0x0", "arg613": "0x0", "arg614": "0x0", "arg615": "0x0", "arg616": "0x0", "arg617": "0x0", "arg618": "0x0", "arg619": "0x0", "arg620": "0x0", "arg621": "0x0", "arg622": "0x0", "arg623": "0x0", "arg624": "0x0", "arg625": "0x0", "arg626": "0x0", "arg627": "0x0", "arg628": "0x0", "arg629": "0x0", "arg630": "0x0", "arg631": "0x0", "arg632": "0x0", "arg633": "0x0", "arg634": "0x0", "arg635": "0x0", "arg636": "0x0", "arg637": "0x0", "arg638": "0x0", "arg639": "0x0", "arg640": "0x0", "arg641": "0x0", "arg642": "0x0", "arg643": "0x0", "arg644": "0x0", "arg645": "0x0", "arg646": "0x0", "arg647": "0x0", "arg648": "0x0", "arg649": "0x0", "arg650": "0x0", "arg651": "0x0", "arg652": "0x0", "arg653": "0x0", "arg654": "0x0", "arg655": "0x0", "arg656": "0x0", "arg657": "0x0", "arg658": "0x0", "arg659": "0x0", "arg660": "0x0", "arg661": "0x0", "arg662": "0x0", "arg663": "0x0", "arg664": "0x0", "arg665": "0x0", "arg666": "0x0", "arg667": "0x0", "arg668": "0x0", "arg669": "0x0", "arg670": "0x0", "arg671": "0x0", "arg672": "0x0", "arg673": "0x0", "arg674": "0x0", "arg675": "0x0", "arg676": "0x0", "arg677": "0x0", "arg678": "0x0", "arg679": "0x0", "arg680": "0x0", "arg681": "0x0", "arg682": "0x0", "arg683": "0x0", "arg684": "0x0", "arg685": "0x0", "arg686": "0x0", "arg687": "0x0", "arg688": "0x0", "arg689": "0x0", "arg690": "0x0", "arg691": "0x0", "arg692": "0x0", "arg693": "0x0", "arg694": "0x0", "arg695": "0x0", "arg696": "0x0", "arg697": "0x0", "arg698": "0x0", "arg699": "0x0", "arg700": "0x0", "arg701": "0x0", "arg702": "0x0", "arg703": "0x0", "arg704": "0x0", "arg705": "0x0", "arg706": "0x0", "arg707": "0x0", "arg708": "0x0", "arg709": "0x0", "arg710": "0x0", "arg711": "0x0", "arg712": "0x0", "arg713": "0x0", "arg714": "0x0", "arg715": "0x0", "arg716": "0x0", "arg717": "0x0", "arg718": "0x0", "arg719": "0x0", "arg720": "0x0", "arg721": "0x0", "arg722": "0x0", "arg723": "0x0", "arg724": "0x0", "arg725": "0x0", "arg726": "0x0", "arg727": "0x0", "arg728": "0x0", "arg729": "0x0", "arg730": "0x0", "arg731": "0x0", "arg732": "0x0", "arg733": "0x0", "arg734": "0x0", "arg735": "0x0", "arg736": "0x0", "arg737": "0x0", "arg738": "0x0", "arg739": "0x0", "arg740": "0x0", "arg741": "0x0", "arg742": "0x0", "arg743": "0x0", "arg744": "0x0", "arg745": "0x0", "arg746": "0x0", "arg747": "0x0", "arg748": "0x0", "arg749": "0x0", "arg750": "0x0", "arg751": "0x0", "arg752": "0x0", "arg753": "0x0", "arg754": "0x0", "arg755": "0x0", "arg756": "0x0", "arg757": "0x0", "arg758": "0x0", "arg759": "0x0", "arg760": "0x0", "arg761": "0x0", "arg762": "0x0", "arg763": "0x0", "arg764": "0x0", "arg765": "0x0", "arg766": "0x0", "arg767": "0x0", "arg768": "0x0", "arg769": "0x0", "arg770": "0x0", "arg771": "0x0", "arg772": "0x0", "arg773": "0x0", "arg774": "0x0", "arg775": "0x0", "arg776": "0x0", "arg777": "0x0", "arg778": "0x0", "arg779": "0x0", "arg780": "0x0", "arg781": "0x0", "arg782": "0x0", "arg783": "0x0", "arg784": "0x0", "arg785": "0x0", "arg786": "0x0", "arg787": "0x0", "arg788": "0x0", "arg789": "0x0", "arg790": "0x0", "arg791": "0x0", "arg792": "0x0", "arg793": "0x0", "arg794": "0x0", "arg795": "0x0", "arg796": "0x0", "arg797": "0x0", "arg798": "0x0", "arg799": "0x0", "arg800": "0x0", "arg801": "0x0", "arg802": "0x0", "arg803": "0x0", "arg804": "0x0", "arg805": "0x0", "arg806": "0x0", "arg807": "0x0", "arg808": "0x0", "arg809": "0x0", "arg810": "0x0", "arg811": "0x0", "arg812": "0x0", "arg813": "0x0", "arg814": "0x0", "arg815": "0x0", "arg816": "0x0", "arg817": "0x0", "arg818": "0x0", "arg819": "0x0", "arg820": "0x0", "arg821": "0x0", "arg822": "0x0", "arg823": "0x0", "arg824": "0x0", "arg825": "0x0", "arg826": "0x0", "arg827": "0x0", "arg828": "0x0", "arg829": "0x0", "arg830": "0x0", "arg831": "0x0", "arg832": "0x0", "arg833": "0x0", "arg834": "0x0", "arg835": "0x0", "arg836": "0x0", "arg837": "0x0", "arg838": "0x0", "arg839": "0x0", "arg840": "0x0", "arg841": "0x0", "arg842": "0x0", "arg843": "0x0", "arg844": "0x0", "arg845": "0x0", "arg846": "0x0", "arg847": "0x0", "arg848": "0x0", "arg849": "0x0", "arg850": "0x0", "arg851": "0x0", "arg852": "0x0", "arg853": "0x0", "arg854": "0x0", "arg855": "0x0", "arg856": "0x0", "arg857": "0x0", "arg858": "0x0", "arg859": "0x0", "arg860": "0x0", "arg861": "0x0", "arg862": "0x0", "arg863": "0x0", "arg864": "0x0", "arg865": "0x0", "arg866": "0x0", "arg867": "0x0", "arg868": "0x0", "arg869": "0x0", "arg870": "0x0", "arg871": "0x0", "arg872": "0x0", "arg873": "0x0", "arg874": "0x0", "arg875": "0x0", "arg876": "0x0", "arg877": "0x0", "arg878": "0x0", "arg879": "0x0", "arg880": "0x0", "arg881": "0x0", "arg882": "0x0", "arg883": "0x0", "arg884": "0x0", "arg885": "0x0", "arg886": "0x0", "arg887": "0x0", "arg888": "0x0", "arg889": "0x0", "arg890": "0x0", "arg891": "0x0", "arg892": "0x0", "arg893": "0x0", "arg894": "0x0", "arg895": "0x0", "arg896": "0x0", "arg897": "0x0", "arg898": "0x0", "arg899": "0x0", "arg900": "0x0", "arg901": "0x0", "arg902": "0x0", "arg903": "0x0", "arg904": "0x0", "arg905": "0x0", "arg906": "0x0", "arg907": "0x0", "arg908": "0x0", "arg909": "0x0", "arg910": "0x0", "arg911":

ALAKEMI at 0x2b4...c3686 (memory)	
addPauser	address account
approve	0xbcaafb4802c27917d29449c8ab707ff6f86193ef, 100
pause	
renouncePauser	
transfer	address to, uint256 value
transferFrom	address from, address to, uint256 value
unpause	
allowance	address _owner, address spender
balanceOf	0x14723a09acff0d2a60ddcf7aa4aff308fddc160c
0: uint256: 12	

- ◆ Verify 4 Wei is added to the global credit status successfully.

getCreditStatus
0: uint256: 1000000000
1: uint256: 1
2: uint256: 10
3: uint256: 999999978
4: uint256: 2
5: uint256: 4

Deposit credits from “ALAKEMI” ERC20 smart contract to messaging contract “AlkemiMessaging”

- ◆ Continue above steps and there are 12 tokens at “ALAKEMI” ERC20 contract and 8 tokens at the messaging contract before the “deposit”

The screenshot shows a web-based Ethereum interface for interacting with the ALAKEMI contract. The top section displays the storage of the ALAKEMI contract at address 0x2b4...c3686 (memory). The storage slots are as follows:

- 0x000: addPauser (address account)
- 0x000: approve (0xbcaafb4802c27917d29449c8ab707ff0f86193ef, 100)
- 0x000: pause
- 0x000: renouncePauser
- 0x000: transfer (address to, uint256 value)
- 0x000: transferFrom (address from, address to, uint256 value)
- 0x000: unpause
- 0x000: allowance (address _owner, address spender)
- 0x000: balanceOf (0x14723a09acff6d2a60dcdf7aa4aff308fddc160c)

The 'balanceOf' slot for address 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c contains the value 12.

The bottom section shows the storage of the AlkemiMessaging contract at address 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c. The storage slots are as follows:

- 0x000: getPlayerInfo (0x14723a09acff6d2a60dcdf7aa4aff308fddc160c)
- 0x000: 0: uint256: 8
- 0x000: 1: uint256: 0
- 0x000: 2: uint8: 1

- ◆ Execute “DepositCredits” transaction and deposit 12 tokens from ERC20 contract to messaging contract

```

115  /* =====Messaging facilitates \ Remove an existing Message(Not allow Reen
116  function removeMessage(uint256 _messageID) external whenNotPaused isHuman n
117  require(_messageID <= messageCount,"Invalid Message ID");
118  require(player[msg.sender].isAuthorized == 1,"This player was not authori
119  require(messageinfo[_messageID].isRemoved == false,"This message already
120  require(player[msg.sender].creditBalance >= COST , "You don't have no eno
121  require(messageinfo[messageCount].Creator == msg.sender , "Only message cr
122
123  player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(C
124
125  creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST); //Upda
126
127  messageinfo[messageCount].isRemoved = true; //Leave message remove flag
128
129  emit evt_MessageAction(msg.sender, _messageID, "Remove");
130  return true;
131 }
132
133 /* =====Credit tokens Operation \ Deposit credit token from ERC20 contrac
134 function depositCredits(uint _amount) external whenNotPaused nonReentrant pa
135  require(depositTokens.transferFrom(msg.sender, address(this), _amount)
136  player[msg.sender].creditBalance = player[msg.sender].creditBalance.a
137  emit evt_TokenAction(msg.sender,_amount, "Deposit");
138  return true;
139

```

[2] only remix transactions, script Search transactions

from	0x14723a09acf8d2a60dcdf7aa4aff308fddc160c
to	AlkemiMessaging.depositCredits(uint256 0xbcaafbf4802c27917d29449c8ab70ff6f86193ef)
gas	5000000 gas
transaction cost	29076 gas
execution cost	36688 gas
hash	0xe769377925196bbef91a0d7fc0f15b38efed3958f7804f197b86c3e63e6d52
input	0xd56...0000c
decoded input	{ "uint256 _amount": "12" }
decoded output	{ "0": "bool: true" }
logs	[{ "from": "0x2b47f0d926a28adfc90a69939502dc687ac3686", "topic": "0xddf25ad1be2c89b69c2b068fc378" }]

Deployed Contracts

ALAKEMI at 0x2b4...c3686 (memory)
AlkemiMessaging at 0xbca...193ef (memory)
(fallback)
addPauser address account
authorizeNewPlaye r 0x14723a09acf8d2a60dcdf7aa4aff308fddc160c
buyCredits
createMessage "Message created by authorized address"
depositCredits 12
kill
pause
removeMessage 1
renouncePauser
unpause
withdrawCredits uint256 _amount
COST
0: uint8: 1
CREDITPRICE
0: uint256: 3
CREDITSONAUTH ORIZED
0: uint256: 10
getCreditStatus
0: uint256: 1000000000
1: uint256: 1
2: uint256: 10

- ◆ Verify the 12 tokens are reduced from ERC20 contract (changed from 12 to 0)

balanceOf 0x14723a09acf8d2a60dcdf7aa4aff308fddc160c

0: uint256: 0

- ◆ Verify the 12 tokens are added to his account at the messaging contract (changed from 8 to 20)

getMessageInfo 1

0: uint256: 1569943808

1: address: 0x14723A09Acff8d2a60DcdF7aA4AfF308FDDC160C

2: string: Message created by authorized address

3: bool: true

getPlayerInfo 0x14723a09acf8d2a60dcdf7aa4aff308fddc160c

0: uint256: 20

1: uint256: 0

2: uint8: 1

isPauser address account

messageCount

0: uint256: 0

owner

0: address: 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c

paused

Withdraw credits from messaging contract “AlkemiMessaging” to “ALAKEMI” ERC20 smart contract

- ◆ Specify 2 tokens and withdraw 2 tokens from messaging contract “AlkemiMessaging” to “ALAKEMI” ERC20 smart contract (Before the transaction, there are 20 tokens balance at his account of messaging contract “AlkemiMessaging”)

The screenshot shows the Ethereum Remix IDE interface. On the left is the Solidity code for the AlkemiMessaging contract, which includes functions for removing messages and depositing credits. On the right is a transaction history table.

```

115  /* ===== Messaging facilitates \ Remove an existing Message(Not allow Reen
116  function removeMessage(uint256 _messageID) external whenNotPaused isHuman n
117  require(_messageID <= messageCount, "Invalid Message ID");
118  require(player[msg.sender].isAuthorized == 1, "This player was not authori
119  require(messageInfo[_messageID].isRemoved == false, "This message already
120  require(player[msg.sender].creditBalance >= COST , "You don't have no eno
121  require(messageInfo[messageCount].Creator == msg.sender , "Only message cr
122
123  player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(C
124
125  creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST); //Upda
126
127  messageInfo[messageCount].isRemoved = true; //Leave message remove flag
128
129  emit evt_MessageAction(msg.sender, _messageID, "Remove");
130  return true;
131 }
132
133 /* ===== Credit tokens Operation \ Deposit credit token from ERC20 contrac
134 function depositCredits(uint _amount) external whenNotPaused nonReentrant pa
135  require(creditTokens.transferFrom(msg.sender, address(this), _amount)
136  player[msg.sender].creditBalance = player[msg.sender].creditBalance.a
137  emit evt_TokenAction(msg.sender,_amount, "Deposit");
138  return true;
139

```

Transactions:

Method	Value
kill	
pause	
removeMessage	1
renouncePauser	
unpause	
withdrawCredits	2
CREDITPRICE	
0: uint8: 1	
CREDITSUPPLY	
0: uint256: 3	
CREDITSUPPLYORIZED	
0: uint256: 10	
getCreditStatus	
0: uint256: 1000000000	
1: uint256: 1	
2: uint256: 10	
3: uint256: 999999978	
4: uint256: 2	
5: uint256: 4	
getMessageInfo	1
0: uint256: 1569943808	
1: address: 0x14723A09ACff6D2A60DcdF7aA4Aff308FDDC160C	
2: string: Message created by authorized address	
3: bool: true	
getPlayerInfo	0x14723a09acff6d2a60dcdf7aa4aff308fdcc160c
0: uint256: 20	
1: uint256: 0	
2: uint8: 1	

- ◆ After the transaction, his account changed from 20 to 2 at the messaging contract “AlkemiMessaging”

```

123     player[msg.sender].creditBalance = player[msg.sender].creditBalance.sub(C
124
125     creditStatus.arisedCredits = creditStatus.arisedCredits.add(COST); //Update
126
127     messageinfo[messageCount].isRemoved = true; //Leave message remove flag
128
129     emit evt_MessageAction(msg.sender, _messageID, "Remove");
130     return true;
131 }
132
133 /* =====Credit tokens Operation \ Deposit credit token from ERC20 contract
134 function depositCredits(uint _amount) external whenNotPaused nonReentrant pa
135     require(creditTokens.transferFrom(msg.sender, address(this), _amount)
136     player[msg.sender].creditBalance = player[msg.sender].creditBalance.a
137     emit evt_TokenAction(msg.sender,_amount, "Deposit");
138     return true;
139 
```

[2] only remix transactions, script Search transactions

decoded output	{ "0": "bool: false"
logs	[{ "From": "0x2b47f0d926a28adfc90a69939502dc b687ac3686", "topic": "0xddf252ad1be2c89b69c2b068fc378 daa952ba7f163c4a11628f5a4df523b3ef", "event": "Transfer", "args": { "0": "0xBcAfb4802C27917D29449c8A B707ff6f86193eF", "1": "0x14723A09ACff6D2A60DcdF7aA 4AFF308FDDC160C", "2": "2", "from": "0xBcAfb4802C27917D29449 c8AB707ff6f86193eF", "to": "0x14723A09ACff6D2A60DcdF7a A4AFF308FDDC160C", "value": "2", "length": 3 } }, { "From": "0xbcaafb4802c27917d29449c8ab707f

unpause	
withdrawCredits	2
COST	
0: uint8: 1	
CREDITPRICE	
0: uint256: 3	
CREDITSONAUTHORIZED	
0: uint256: 10	
getCreditStatus	
0: uint256: 1000000000	
1: uint256: 1	
2: uint256: 10	
3: uint256: 999999978	
4: uint256: 2	
5: uint256: 4	
getMessageInfo	1
0: uint256: 1569943808	
1: address: 0x14723A09ACff6D2A60DcdF7aA4AFF308FDDC160C	
2: string: Message created by authorized address	
3: bool: true	
getPlayerInfo	0x14723a09acf0d2a60dcd7aa4aff308fddc160c
0: uint256: 18	
1: uint256: 0	
2: uint8: 1	

- ◆ Verify the 2 tokens are transferred to his account at the messaging contract "AlkemiMessaging"

```

130     return true;
131 }
132
133 /* =====Credit tokens Operation \ Deposit credit token from ERC20 contract
134 function depositCredits(uint _amount) external whenNotPaused nonReentrant pa
135     require(creditTokens.transferFrom(msg.sender, address(this), _amount)
136     player[msg.sender].creditBalance = player[msg.sender].creditBalance.a
137     emit evt_TokenAction(msg.sender,_amount, "Deposit");
138     return true;
139 
```

[2] only remix transactions, script Search transactions

decoded output	{ "0": "bool: false"
logs	[{ "From": "0x2b47f0d926a28adfc90a69939502dc b687ac3686", "topic": "0xddf252ad1be2c89b69c2b068fc378 daa952ba7f163c4a11628f5a4df523b3ef", "event": "Transfer", "args": { "0": "0xBcAfb4802C27917D29449c8A B707ff6f86193eF", "1": "0x14723A09ACff6D2A60DcdF7aA 4AFF308FDDC160C", "2": "2", "from": "0xBcAfb4802C27917D29449 c8AB707ff6f86193eF", "to": "0x14723A09ACff6D2A60DcdF7a A4AFF308FDDC160C", "value": "2", "length": 3 } }, { "From": "0xbcaafb4802c27917d29449c8ab707f

ALAKEMI at 0x2b4...c3686 (memory)	
addPauser	address account
approve	0xbcaafb4802c27917d29449c8ab707ff6f86193eF,12
pause	
renouncePauser	
transfer	address to, uint256 value
transferFrom	address from, address to, uint256 value
unpause	
allowance	address _owner, address spender
balanceOf	0x14723a09acf0d2a60dcd7aa4aff308fddc160c
0: uint256: 2	
isPauser	address account
name	
0: string: ALAKEMI	
owner	0: address: 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c
paused	