Registration Agreement

1. Overview

Issuer	Smart Contracts Lab, Zurich, Switzerland, ("Issuer" or "Company")
System	Polygon Mainnet ("Blockchain"), see https://polygon.technology/ for more information
Register	Smart-Contract: 0x9a039C5a20D89DAB69C8B12C53E55DE2464E5Cf3 on the Polygon Mainnet ("Token Contract" or "Token Register"), named "SCL_Equity_Token" with symbol "SCL"
Source code	Available on Polygonscan[1] and GitHub[2]
Issued Instrument	Equity Shares of the Issuer ("Shares")
Number of Shares outstanding and their nominal value	See the commercial registry[3]
Additional source of information	https://ce-tokenization.vercel.app/ ("Website")
Transfer restrictions	Currently none

[1] https://mumbai.polygonscan.com/address/0x9a039C5a20D89DAB69C8B12C53E55DE2464E5Cf3
[2] https://github.com/SCL-Project/Tokenization/professor/contract/Swiss Equity Token 1.9.sol
[3] https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

2. Legal Framework

The board of directors is allowed by Article 4a of the articles of association[4] to change the legal form of the Shares upon request of the respective Shareholder, including the conversion of Shares held as uncertificated securities according to Article 973c of the Swiss Code of Obligations ("CO") into register securities according to Article 973d CO. Paragraph 1 states, that a ledger-based security is a right which, according to an agreement of the parties is registered in a securities ledger in accordance to Article 973d paragraph 2, and can only be asserted and transferred to others via this securities ledger. Each share that is issued as a register value right (“Share Tokens”) is assigned one numeric unit in the token contract. This Registration Agreement (the "Registration Agreement" or "Agreement") sets out the terms relating to the Share Tokens. The Issuer and all Token Holders are bound to this Agreement by law. The Issuer reserves the right to modify the Registration Agreement at any time to reflect the latest legal and technical developments, as well as decisions taken by the general assembly, or the board of directors as tasked by the general assembly[5]. Where necessary, the Company informs in an appropriate way the registered Shareholders about changes in accordance with the articles of association.

3. Share Register

3.1 Shareholder Registration

Additional to the Token Register the Company keeps a separate off-chain share register (the "Share Register"). Any legal person or entity, which can demonstrate the power to dispose of one or more tokens (a "Token Holder") can send a registration request per electronic form provided by the Company via Website[6]. Only the subjects registered in the Share register are entitled to Shareholder rights (such as voting and dividend rights). Token Holders hereby acknowledge and consent to registering themselves as shareholders with the Issuer shortly after acquiring or receiving their Share Tokens. In addition, they acknowledge and agree to notify the Issuer of any change in circumstances relevant for the registration within a period of 30 calendar days.

The Token Holder must disclose personal information and supply sufficient proof of possessing and controlling the account in which the Share Tokens in question are held to request the entry to the Share register. The registration form includes an option to enable recoverability for the event of a potential loss of the Token Holder’s private key. By opting for this option, the Token Holder consents to grant partial authority over the account to the Issuer and Deputy.

---

[4] Available under: https://xxxxxxxxxxxxx
[5] The latest version is available on our Website
[6] Registration form available under: https://ce-tokenization.vercel.app/

For possible tokens held through an intermediary, the intermediary may perform the registration on behalf of the beneficiary.

3.2 Change of beneficial ownership

If a change in the beneficial ownership of registered Token Holders occurs, the new beneficial owners are required to report those changes to the Company. If a third party holds Share Tokens of different beneficial owners on the same account, the third party is required to report to the Company the Share Tokens of which beneficial owner has been transferred. In absence of any indication, the Company will assume that Share Tokens acquired first are also disposed of first.[7]

3.3 Transfer Restrictions

The Company may refuse entry into the Share register in accordance with the transfer restrictions in the articles of association. In case the Company refuses the entry of a Token Holder into the Share register, such Token Holder can validly transfer the affected Share Tokens regardless of the admission into the Share register. As of the date of this Registration Agreement, there are no transfer restrictions. For further information about existing constraints, please contact the Issuer and request a copy of the articles of association.

3.4 Sub-Registers

The Company is free to recognize sub-registers. A sub-register can consist of any technical measure to keep track of the ownership of a multitude of Share Tokens assigned to a single address in the Token Register. It could, for example, be another smart contract or a register deployed to a different blockchain. In case the Company enables a sub-register, it suffices for the Token Holder to prove control of the sub-register (or a fraction thereof) to be registered as a Shareholder for the tokens held by the sub-register (or the respective fraction thereof).

---

[7] This procedure is called FIFO

3.5 Fractionability

Smart Contracts Lab enables the transfer and holding of fractional units of a token. "Balance" is the sum of "Shares" and "Fractions". To determine the "Shares" assigned to the respective Token Holder, the "Balance" is rounded down to the next integer. As soon as "Fractions" becomes a whole number, it is converted to a Share.
Fractional tokens are the rightful property of the Issuer. Hence, these fractional parts of tokens do not entitle the Token Holder to Shareholder rights (such as voting and dividend rights).

3.6 Updates, Changes, and Delegation

The Issuer may update the Token Contract unilaterally as reasonably required for regulatory or technical reasons. He is also authorized to execute a change in the Blockchain for said reasons. Tasks, such as updating the Share register and validating requests, may be delegated to a third party. Any changes in the personal data of the Token Holder, such as name changes and/or changes of address must be immediately notified to the Issuer, or the third party designated by the Issuer.

3.7 Establishment of Securities

According to Article 973g paragraph 1 number 1 CO, the registration of securities is technically not supported in the Token Register. To legally establish a security on a Share token, the transfer of the Share token or the use of a sub-register that offers this required functionality is required.

4. Hard Fork

The Company has complete discretion in case of a split of the Blockchain (hard fork) to decide which version of the Blockchain will be considered to hold the real Share Tokens. In such an occurrence, the Company will communicate its decision publicly on the Website[8].

[8] https://ce-tokenization.vercel.app/

5. Transfer

The token contract makes it possible for Token Holders to transfer tokens from their blockchain address to another blockchain address, provided that each transaction of previously registered Share Tokens has been notified to the Company and delisted from the Share register.

5.1 Legal Principle

A legally valid transfer of a Share token is constituted by any action that succeeds in the shift of the capability to demonstrate the power to dispose of a Share Token. Examples include the transfer of private keys to a new owner by simple transaction or sending a paper wallet by mail.

5.2 Inseparability

Shares issued as Share Tokens are indivisibly bound to each other in accordance with Article 973d CO. Thus, transferring a Share Token without including the right to register a Share in the Share register and vice versa is not possible. According to the abstractness effect, the transfer of Share Tokens is, in any case, always legally effective irrespective of the validity of the underlying obligatory transaction. No grounds for invalidity, such as lack of will, material error, or withdrawal of consent to the transfer, may be invoked.
As per Article 1006 CO, the acquirer of Share Tokens is protected in the acquisition even if the transferor did not have the right to dispose of the Share Tokens, an exception is made if gross negligence or bad faith is involved. Furthermore, in case of bankruptcy, seizure of assets, or a moratorium of a Token Holder, Article 973f CO applies.

6. Raise

The “Raise” function raises the total amount of Shares issued in the smart contract. This ability can be executed by the Company if and only if:

i. the general assembly approves a Share capital increase resulting in a statuary modification in accordance with Articles 650 and 651 CO

7. Pause Function

Additionally, the smart contract possesses the possibility to freeze the tokens to prevent the execution of transactions on the blockchain. The "Pause" function will be put into practice by the Company if and only if:

i. a hard fork occurs during the decisional time as to which version of the blockchain the Company will support or
ii. the Company is enforced to execute a court ruling in relation to the violation of the anti-money laundry directive

The "Unpause" will revert the inability to transfer tokens.

8. Loss of Tokens

According to Article 973d CO, the Company is prohibited from harming the integrity of the token contract with changes of any kind. Nevertheless, the Company can unilaterally execute transactions using a dedicated function (recovery function) on its token contract. Token Holders that have lost access to their Share Tokens can submit a request to reclaim them if and only if they have chosen the recoverability option during the registration. The Company is allowed to use the recovery function if and only if one of the afterward named prerequisites is given:

i. the execution of a juridical order
ii. helping Token Holders who lost their private key (under the circumstances that they can proof their ownership in accordance with the Company's complete requirements)

In case of recovery, the allowances associated with the old address will become invalid.

Alternatively, a judge can declare Share Tokens invalid following the procedure represented in Article 973h CO. The Company will replace those with new Share Tokens and, in full discretion, decide whether to destroy the invalid Share Tokens or leave them in the Token Register and, according to this procedure, inform all Shareholders on the Website[9].

[9] https://ce-tokenization.vercel.app/

9. Rejection of Accountability

Herewith Smart Contracts Lab excludes, to the extent permitted by law, all representations and warranties with regard to the Shares and the Share Tokens and any liability by the Company or any person acting on behalf of the Company.
Any and all claims related to misstatements or breaches of warranties it may have under the applicable law will hereby be explicitly waived by the Token Holder.

10. Tax Implications

It is the sole responsibility of the Token Holder to determine if its purchase, the potential appreciation, or depreciation in the Share Tokens over time, the sale of Share Tokens, and/or any other action or transaction related to the Share Tokens lead to tax implications for the Token Holder.

11. General Arrangements

11.1 Severability/ Good Faith

Should any competent court, governmental or administrative authority having jurisdiction (in the area Smart Contracts Lab operates) hold any part or provision of this Agreement to be invalid, the other provisions of this Agreement shall nonetheless remain valid. The Company will in such a case provide a substitute that best reflects the economic intentions without being unenforceable and will execute all agreements and provide all documents required in this connection. If and to the extent that this Agreement is to contain any gaps or possible omissions, the same procedure shall apply.

11.2 Reigning Law and Jurisdiction

This Agreement was construed in accordance with the substantive laws of Switzerland and will be implemented thereby as Article 145a PILA. All disputes arising out of or in connection with the present agreement, including disputes on its conclusion, binding effect, and termination, shall be resolved by the ordinary courts in Zurich, Switzerland.

12. Risk Considerations

12.1 Common Risks

The investment in Share Tokens may offer an opportunity for capital gains but also entail a high degree of business and financial risks, including the possibility of a complete loss of the investment.
The purchase or sale of Share Tokens is not solicited, by any means, in this document. Smart Contracts Lab requests instead each purchaser to engage in his own independent research and make his own decisions with respect to the purchase of Share Tokens.
To evaluate the implied risks, the Company also recommends, in cases of lack of knowledge, to consult a lawyer, an accountant, and/or a tax professional.

Further, the risks described herein are not the only risks that come into question and are by no means intended to represent a comprehensive list. Buying Share Tokens may expose potential purchasers also to other risks of other nature. Any indication of the probability of occurrence or seriousness or importance of the individual risks or their impact in the event they are occurring, is in no case provided by the order in which the individual risks were chosen to be exposed in this document.
Non-business-specific risks that are not yet known by the Company or that the Company does not currently interpret to be relevant may as well occur and have an impact on the Token Holders.

A comprehensive understanding of the Share token's nature and their degree of risk exposure in connection with the individual risk capability of every Token Holder is of vital importance. This, because the investment in Share Tokens should be suitable, taking into consideration the own circumstances and financial condition. Therefore, Token Holders should ensure themselves that they have fully understood this concept.

In respect of everything mentioned above, the Share Tokens involve a high degree of risk, including the potential risk of expiring worthless. A potential purchaser must be prepared in certain circumstances to sustain a total loss of the capital invested in purchasing Share Tokens.

12.2 Regulatory Risks

Blockchain technology allows new forms of decentralized interaction between financial parties and offers even more potential in the future regarding the efficiency and security of financial transactions. Thus, it is possible that certain jurisdictions will apply existing regulations on or introduce brand new laws addressing blockchain technology-based applications, which may result in a contradiction to the current setup of the Share Tokens and their current legal

framework. This may result in forced substantial modifications of the Share Tokens, including their potential loss or invalidation.

12.3 Operational and IT Risks

Blockchain technology, in general, and the smart contract concept on which the Share Tokens are built are both still in an early development stage and unproven. Therefore, there is no warranty that the process of creating, receiving, holding, and storing Share Tokens will be uninterrupted or impeccable, and there is an inherent risk that the software could contain weaknesses, vulnerabilities, or bugs, causing inter alia a complete loss of the Share Tokens. In addition, activities that could result in the theft or loss of the Share Tokens, such as hacking attacks, are possible and cannot be completely excluded. The theft or loss of the Issuer's or Deputy's private key would be publicly communicated on the Website. In the event of a loss of the Issuer's or Deputy's private key, the smart contract would no longer be legally valid.
By enabling the recoverability option, the Token Holder acknowledges that in the event of the Issuer's or Deputy's private key being stolen, there is a possibility of misuse of the recovery function. Furthermore, the Issuer has only limited influence on the proper functioning of the smart contract since it may be subject to future changes and unforeseen problems.

In particular, blockchains are vulnerable when "mining attacks" occur, including but not limited to so-called "51%-attacks", "selfish-mining attacks", timestamp manipulation, and/or race condition attacks. Although Smart Contracts Lab will attempt with all the possible precautions to avoid such an undertaking, any successful attack presents a risk to the Token Holder by not enabling proper execution and sequencing of transactions as well as not enabling proper execution of contract stipulations. This may potentially result in the loss or malfunction of Share Tokens.

Using third-party smart contracts can be an option when the desire arises to autonomously manage Share Tokens. This could lead, depending on the precise implementation, to a situation where a malicious claim on the Share Tokens held by the contract address cannot be cleared by the rightful owner. It is crucial to notice that the liability for the loss of tokens resulting from incompatible implementation of a third-party smart contract cannot be held by the Company.

12.4 Loss of Keys

If the respective private key to a wallet is lost or due to malfunctioning or incompatibilities of the wallet in which the Share Tokens are stored, Share Tokens can become inaccessible. It can represent another risk also leading to the loss of the Share Tokens. In accordance with the law, it is the responsibility of the Token Holder not to lose the key or password that allows access to their personal wallet.

13. Signatures

In accordance with the articles of association, the board of directors herewith declares that this Registration Agreement governs all Share Tokens in accordance with Article 973d CO.

PLACE, DATE

Smart Contracts Lab
Christian Ewerhart
President of the Board of Directors

14. Appendix

```
// ****************************************
// SPDX-License-Identifier: MIT
// Swiss Equity Token 1.9
// Author: Smart Contracts Lab, UZH
// Created: June 29, 2023
// ****************************************

pragma solidity ^0.8.19;

interface IERC20 {
    event Transfer(address indexed from, address indexed to, uint value);
    event Approval(address indexed owner, address indexed spender, uint
value);
    function totalSupply() external view returns (uint);
    function balanceOf(address account) external view returns (uint);
    function transfer(address to, uint amount) external returns (bool);
    function allowance(address owner, address spender) external view re-
turns (uint);
    function approve(address spender, uint amount) external returns (bool);
    function transferFrom(address from, address to, uint amount) external
returns (bool);
}

contract SwissEquityToken is IERC20 {

    event Registered(address indexed account, bytes32 hash, bool recovera-
ble);
    event Recovered(address indexed oldAccount, address indexed
newAccount);

    struct Investor {
        uint balance;
        uint shares;
        uint fractions;
        bool known;
        uint ID;
        bytes32 hash;
        bool recoverable;
    }

    mapping(address => Investor) private _registry;
    mapping(address => mapping(address => uint)) private _allowance;

    string private _name;
    string private _symbol;
    uint8 private immutable _decimals;
    uint private _totalSupply;

    uint public _totalShares;
    address public _issuer;
    address public _deputy;
    address[] public _investors;
    uint public immutable _ONE_SHARE;
    uint public _treasuryShares; // owned by issuer but inaccessible
    bool public _paused;

    modifier onlyIssuer() {
```

```
        require((msg.sender == _issuer) || (msg.sender == _deputy), "only
issuer");
        _;
    }

    constructor() {
        _name = "Swiss Equity Token";
        _symbol = "SET";
        _decimals = 18;
        _ONE_SHARE = 10 ** _decimals;
        _totalShares = 10000000;
        _totalSupply = _totalShares * _ONE_SHARE;
        _issuer = msg.sender;
        _deputy = 0x41EaC9c0E5EA02ae690f37CdA6fB1cdDECD752b1;
        _registry[_issuer].balance = _totalSupply;
        _registry[_issuer].shares = _totalShares;
        _registry[_issuer].known = true;
        _investors.push(_issuer);
    }

// ************************ ERC20 Module ************************

    function name() public virtual view returns (string memory) { return
_name; }

    function symbol() public virtual view returns (string memory) { return
_symbol; }

    function decimals() public virtual view returns (uint8) { return _deci-
mals; }

    function totalSupply() public virtual view returns (uint)  { return
_totalSupply; }

    function balanceOf(address owner) public virtual view returns (uint) {
return _registry[owner].balance; }

    function transfer(address to, uint value) public virtual returns (bool)
{
        settleTransfer(msg.sender, to, value);
        return(true);
    }

    function allowance(address owner, address spender) public virtual view
returns (uint) { return _allowance[owner][spender]; }

    function approve(address spender, uint value) public virtual returns
(bool) {
        _allowance[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }

    function transferFrom(address from, address to, uint value) public vir-
tual returns (bool) {
        require(value <= _allowance[from][msg.sender],"allowance too low");
        settleTransfer(from, to, value);
        _allowance[from][msg.sender] -= value;
        return true;
    }
```

```
// ************************ Shareholder Module ************************

    function sharesOf(address owner) public view returns (uint) { return
_registry[owner].shares; }

    function fractionsOf(address owner) public view returns (uint) { return
_registry[owner].fractions; }

    function transferShares(address to, uint shares) public returns (bool)
{
        settleTransfer(msg.sender, to, shares * _ONE_SHARE);
        return(true);
    }

    function register(bytes32 hash, bool recoverable) public {
        require(_registry[msg.sender].known);
        _registry[msg.sender].hash = hash;
        _registry[msg.sender].recoverable = recoverable;
        emit Registered(msg.sender, hash, recoverable);
    }

// ************************ Settlement Module ************************

    function settleTransfer(address from, address to, uint value) internal
{
        require(_paused == false, "paused");
        require(value <= _registry[from].balance, "balance too low");

        uint _shares = value / _ONE_SHARE;
        uint _fractions = value % _ONE_SHARE;
        uint _checkSum = _treasuryShares + _registry[from].shares + _regis-
try[to].shares;

        _registry[from].balance -= value; // debit
        _registry[to].balance += value; // credit

        if (_fractions > _registry[from].fractions) { // insufficient frac-
tions to settle debit: one share is swapped into fractions
            _registry[from].shares -= 1;
            _treasuryShares += 1;
            _registry[from].fractions += _ONE_SHARE;
        }

        _registry[from].shares -= _shares;
        _registry[to].shares += _shares;
        _registry[from].fractions -= _fractions;
        _registry[to].fractions += _fractions;

        if (_registry[to].fractions >= _ONE_SHARE) { // too many fractions
resulting from credit: swap is reversed
            _registry[to].shares += 1;
            _treasuryShares -= 1;
            _registry[to].fractions -= _ONE_SHARE;
        }

        assert(_registry[from].shares == _registry[from].balance /
_ONE_SHARE);
        assert(_registry[from].fractions == _registry[from].balance %
_ONE_SHARE);
```

```
        assert(_registry[to].shares == _registry[to].balance / _ONE_SHARE);
        assert(_registry[to].fractions == _registry[to].balance %
_ONE_SHARE);
        assert(_checkSum == _treasuryShares + _registry[from].shares +
_registry[to].shares);

        if (_registry[to].known == false) {
            _investors.push(to);
            _registry[to].ID = _investors.length;
            _registry[to].known = true;
        }

        emit Transfer(from, to, value);
    }

// ************************ Issuer Module ************************

    function numberOfInvestors() public view returns(uint) { return _inves-
tors.length; }

    function pause() public onlyIssuer { _paused = true; }

    function unpause() public onlyIssuer { _paused = false; }

    function recover(address oldAccount, address newAccount) public onlyIs-
suer {
        require(_registry[oldAccount].recoverable || oldAccount == _issuer,
"not recoverable");
        require(_registry[newAccount].known == false, "in use");

        _registry[newAccount] = _registry[oldAccount];
        _investors[_registry[newAccount].ID] = newAccount;

        if (oldAccount == _issuer) {_issuer = newAccount;}
        if (oldAccount == _deputy) {_deputy = newAccount;}

        delete _registry[oldAccount];
        emit Recovered(oldAccount, newAccount);
    }

    function raise(uint shares) public onlyIssuer {
        uint _value = shares * _ONE_SHARE;
        _registry[_issuer].balance += _value;
        _registry[_issuer].shares += shares;
        _totalSupply += _value;
        _totalShares += shares;
        emit Transfer(address(0), _issuer, _value);
    }

    function changeDeputy(address deputy) public onlyIssuer { _deputy =
deputy; }

}
```