SDS SETTING DIGITAL STANDARDS

ZWISCHENVERWAHRUNG DIGITALER ASSETS

GEMEINSAMKEITEN UND UNTERSCHIEDE ZUR TRADITIONELLEN GIROSAMMELVERWAHRUNG AUS TECHNISCHER SICHT

VIENNA FINANCIAL DATA SUMMIT 2025

www.sds.at

VERSION XX.XX

Braucht der Intermediär Krypto

Braucht Krypto den Intermediär

KLASSEN DIGITALER ASSETS

ART / EMT

- Auf einen (Vermögens)-Wert oder eine amtliche Währung wertreferenzierte Tokens (z.B. Stablecoins, E-Geld-Tokens)

Payment Tokens ohne Referenz

- Vulgo „Kryptowährungen“
- In MiCAR als „andere Kryptowerte“ eingestuft

Sonstige „andere Kryptowerte“

- Andere Kryptowerte als wertreferenzierte Tokens und E-Geld-Tokens nach MiCAR

Security Tokens

- Wertpapier- oder wertpapierähnliche Tokens

Elektronisches Wertpapier

- Elektronisch begeben
- Weist ansonsten die Merkmale eines „klassischen“ Wertpapiers auf

ROLLE DES INTERMEDIÄRS

- Verwahrung
- Auslösen von bestandsverändernden Transaktionen
- Durchführung von Zahlungen und Corporate Actions
- Schlüsselverwaltung
- Erfüllung regulatorischer Auflagen
  aus MiFID und MiCAR

STAMMDATEN

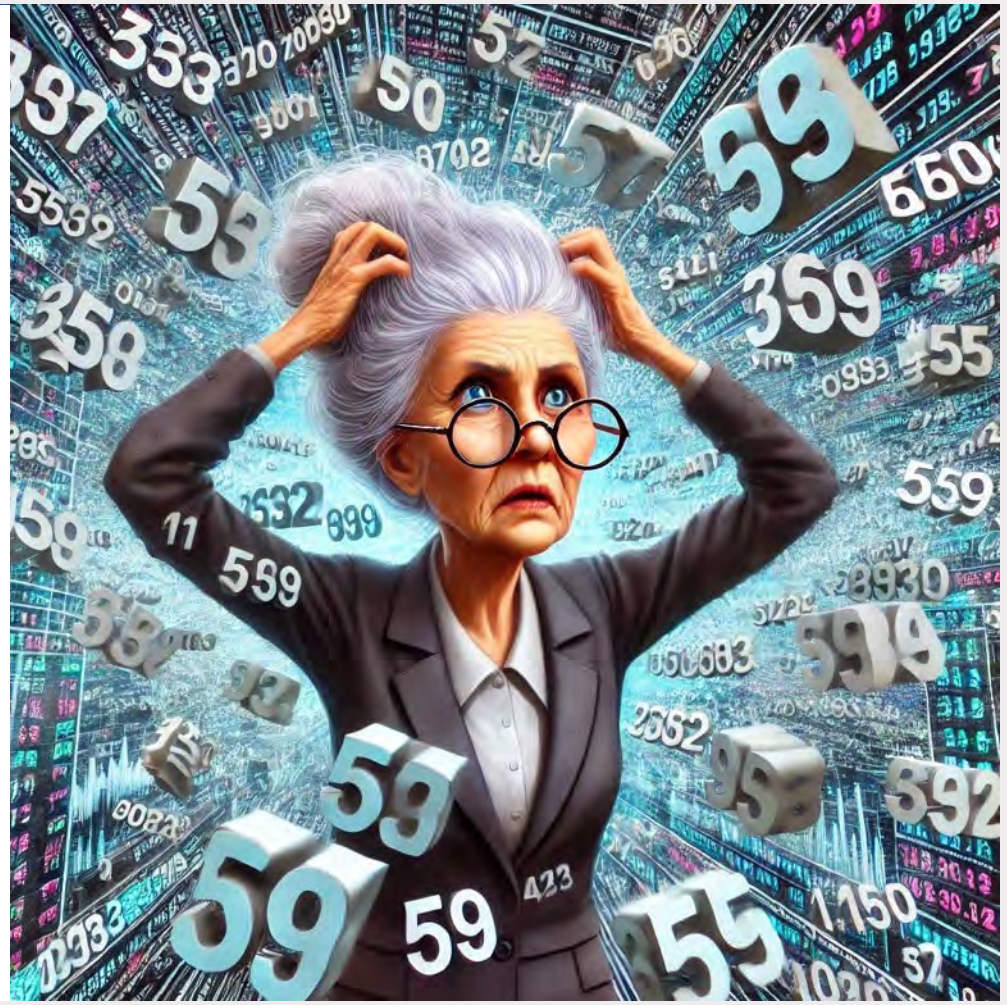
- Kennnummern
- Technische Eigenschaften
- Regulatorische Klassifikation
- Emittenteninformation

- Beispiel:
  - WM-Produkt ITR
  - WM-Produkt DAM

Feldident	Feldbezeichnung
AD003	Wirtsch.Zweck ITC
AZ004	Branche Emi. ITC
AD005	Tech. Dimen. ITC
AD006	LEGAL CLAIM ITC
AD007	Emi. Typ ITC
AD008	STATUS MICA ITC
AD009	Emi-Name Token
AD010	Stat. DatBr. ITSA
ZD011	TokenImplement.
AD012	TokenFungibil.
AD013	Ledgers des Token
AD014	Art des Ledgers
AD015	HashFunk. Ledger
AD016	GenBlockHashLedg
AD017	GenBlockNumLedg
AD018	GenBlockTimStLedg

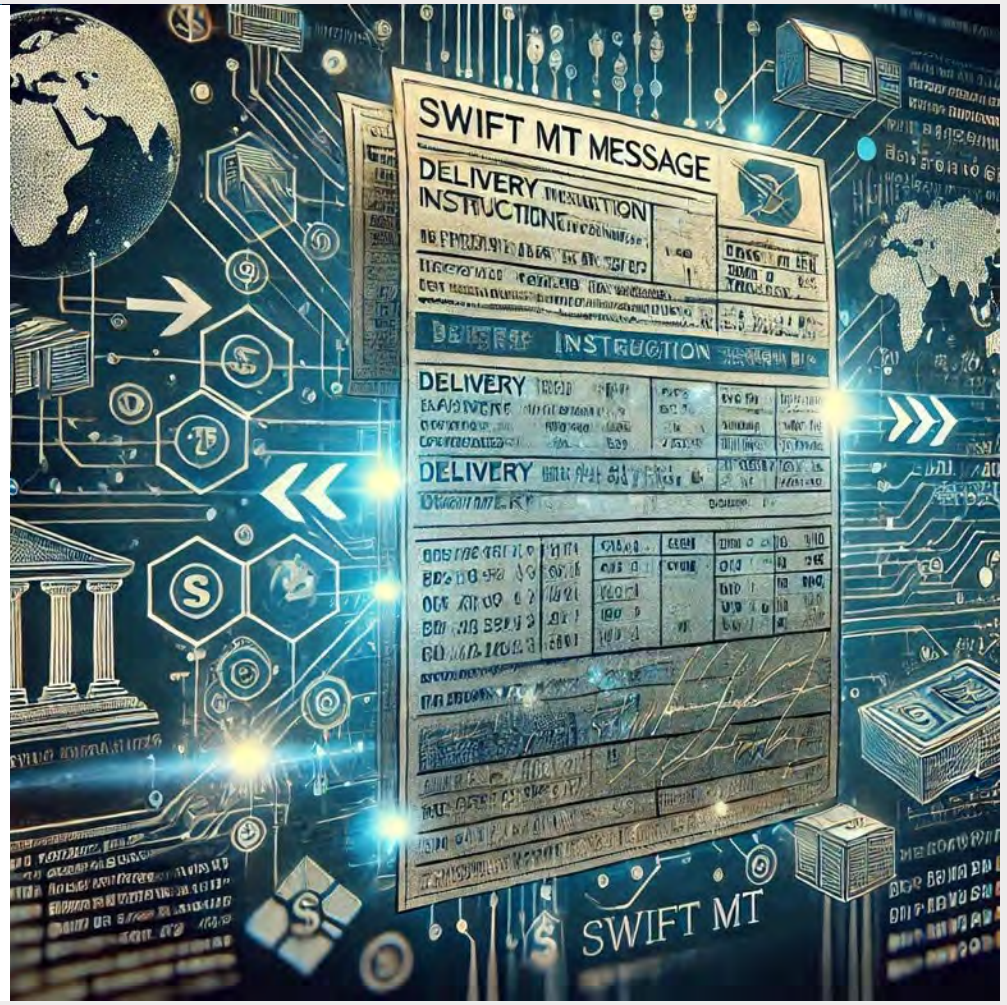
POSITIONSDARSTELLUNG

- Mengenangaben und kleinste Stückelung:
  - Unkonventionelle Einheiten
    - BTC: 20.999.999,97690000 (16 Stellen)
    - 1 ETH =1 000 000 000 000 000 000 Wei (19+ Stellen)
  - Unkonventionelle Kurse:
    - 1 SHIB ~ 0,0000138 USD
    - 1 BTC ~ 100.000 USD
  - SWIFT: bis zu 15 Vor- und Nachkommastellen, in der Praxis fast immer weniger
  - Domänenänderungen bzw Datenkonvertierungen in zahlreichen Systemen nötig
- Umgehungsmöglichkeiten abhängig vom Geschäftsmodell



INSTRUKTIONEN

- Lieferinstruktionen zur Positionsbildung
  - Nutzung weiterer Intermediäre:
    - Ermöglicht unter Einschränkungen konventionelle Instruktionen (SWIFT 15022, 20222)
    - Abrechnungen in Fiatwährung
    - Weitgehende Nutzung bestehender Infrastruktur
  - Instruktionen direkt an eine Blockchain
    - DLT-spezifische Implementierung der Kommunikation
    - Autorisierung über Schlüssel („your key, your coins“)
    - Mapping von Adressen und Depotnummern
    - Besondere Compliance-Vorschriften



ZAHLUNGEN UND CORPORATE ACTIONS

- Codierung durch Smart Contracts
- Zwischenschaltung eines Dienstleisters oder direkte Kommunikation mit dem Smart Contract
- Smart Contracts codieren Verträge, insbesondere mit Bedingungen, unter denen sie bestimmte Transaktionen auslösen.
- Wechsel von einer nachrichtenbasierten Welt in eine an Funktionen orientierte Welt.

BEISPIEL EINES EINFACHEN SMART CONTRACT

- Anleihe auf Ethereum:
  - Der Smart Contract ist eine Art Konto mit weiteren Funktionen.
  - Die weiteren Funktionen können wir selbst codieren.
  - Jemand der den Smart Contract verwenden will, kann darauf buchen. Dabei können oder müssen die weiteren Funktionen ausgelöst werden.
- Der Contract bietet einfache Möglichkeiten:
  - Einzahlung durch Investoren
  - Entnahme durch den Emittenten
  - Rückzahlung samt Zins

SDS

Blockchain: Ethereum, Sprache: solidity

BASISPARAMETER

EREIGNISSE

BESCHRÄNKUNGEN

ERSTELLEN DES SMART CONTRACT

```
contract BondContract {
    address public immutable issuer;
    uint public immutable endInvestmentDate;
    uint public immutable maturityDate;
    uint public immutable interestRate;
    uint public immutable maxInvestment;

    mapping(address => uint) public investments;
    address[] public investors;

    event Invested(address investor, uint amount);
    event Payout(address investor, uint amount);
    event Funded(uint amount);
    event Withdrawn(uint amount);

    modifier onlyIssuer() {
        require(msg.sender == issuer, "Nur der Emittent kann dies tun");_;}

    modifier onlyDuringInvestmentPhase() {
        require(block.timestamp <= endInvestmentDate, "Investitionsphase beendet");_;}

    modifier onlyAfterMaturity() {
        require(block.timestamp >= maturityDate, "Anleihe noch nicht faellig");_;}

    constructor(uint _interestRate, uint _maxInvestment) {
        issuer = msg.sender;
        endInvestmentDate = block.timestamp + 10 days;
        maturityDate = endInvestmentDate + 365 days;
        interestRate = _interestRate;
        maxInvestment = _maxInvestment;
    }
```

ERZEUGT LISTE MIT DEN EINZELNEN INVESTMENTS

```
function invest() public payable onlyDuringInvestmentPhase {
    require(address(this).balance <= maxInvestment, "Maximale Investition erreicht");
    require(msg.value > 0, "Investition muss groesser als 0 sein");

    if (investments[msg.sender] == 0) {
        investors.push(msg.sender);
    }

    investments[msg.sender] += msg.value;
    emit Invested(msg.sender, msg.value);
}
```

EMITTENT KANN ABBUCHEN

```
function withdrawFunds(uint amount) public onlyIssuer {
    payable(issuer).transfer(amount);
    emit Withdrawn(amount);
}
```

EMITTENT KANN EINZAHLEN

```
function fundContract() public payable onlyIssuer {
    require(msg.value > 0, "Muss eine Einzahlung leisten");
    emit Funded(msg.value);
}
```

SDS

```
function payout() public onlyAfterMaturity {
    uint totalPayout = 0;
    // Berechne die insgesamt gezeichnete Menge und den Zins:
    for (uint i = 0; i < investors.length; i++) {
        address investor = investors[i];
        uint investedAmount = investments[investor];

        uint payoutAmount = investedAmount + (investedAmount * interestRate / 100);
        totalPayout += payoutAmount;



    }
    //Prüfe, ob der Emittent ausreichend Mittel zur Verfügung gestellt hat;
    require(address(this).balance >= totalPayout, "Nicht genug Guthaben fuer Auszahlung");

    //Auszahlung an alle Investoren
    for (uint i = 0; i < investors.length; i++) {
        address investor = investors[i];
        uint investedAmount = investments[investor];

        if (investedAmount > 0) {
            uint payoutAmount = investedAmount + (investedAmount * interestRate / 100);
            investments[investor] = 0;
            payable(investor).transfer(payoutAmount);
            emit Payout(investor, payoutAmount);
        }
    }
}
```

ZAHLUNGEN UND CORPORATE ACTIONS

- Codierung durch Smart Contracts
- Zwischenschaltung eines Dienstleisters oder direkte Kommunikation mit dem Smart Contract
- Smart Contracts codieren Verträge, insbesondere mit Bedingungen, unter denen sie bestimmte Transaktionen auslösen.
- Von der nachrichtenbasierten (SWIFT-)Welt in eine an Funktionen orientierte Welt:
  - Als Emittent muss man Smart Contracts erstellen, testen und publizieren.
  - Als Intermediär muss man Funktionen im eigenen oder im Namen von Kunden aufrufen und Ergebnisse ggf weiterverarbeiten.

REGULATORISCHE ANFORDERUNGEN

- Pflichtbelege wie Depotauszug und Kostentransparenz
- Steuerberechnung und – einbehalt
- Eigenbestand
- Meldewesen, z.B.:
  - Transaktionsmeldungen
  - CARF Reporting


Stammdaten
Positionsdarstellung
Handel und Settlement
Corporate Actions
Regulatorik
Generated by GPT-40

SDS