

Report on the DLT Pilot Regime

Study on how financial instrument transactions are registered in various Distributed Ledger Technologies

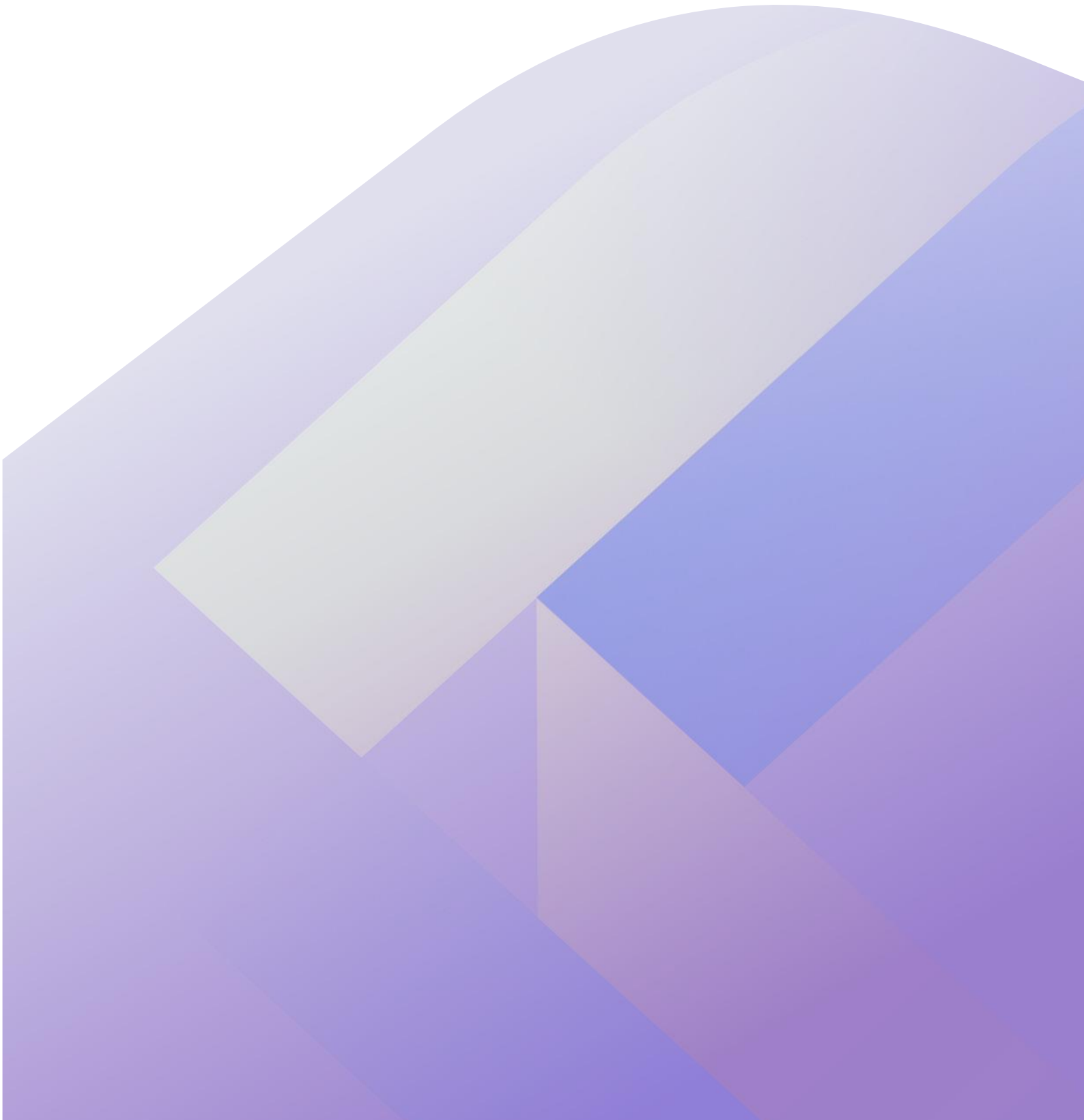


Table of Contents

Acronyms used.....	4
1 Executive Summary	7
2 Introduction to the DLT Pilot and MiFID II/MiFIR transaction reporting	9
3 Registration of transactions on selected Distributed Ledger Technologies	11
3.1 Corda	11
3.1.1 Background.....	11
3.1.2 Applied methodology.....	13
3.1.3 Main elements of a Corda transaction	14
3.1.3.1 Definition of a Corda transaction.....	14
3.1.3.2 Types of Corda transactions	18
3.1.3.3 Established trading processes compared to DLT trading processes	21
3.1.3.4 Structure of a Corda transaction	24
3.1.4 Gap analysis with respect to RTS 22.....	36
3.1.4.1 Fields similar to RTS 22.....	37
3.1.4.2 Fields not covered in RTS 22.....	38
3.1.4.3 Fields of particular importance	38
3.1.4.4 Fields relevant for on-chain analysis	41
3.1.5 Conclusion	42
3.2 Ethereum	43
3.2.1 Background.....	43
3.2.2 Applied methodology.....	47
3.2.3 Main elements of an Ethereum transaction	48
3.2.3.1 Definition of an Ethereum transaction	48
3.2.3.2 Types of Ethereum transactions	48
3.2.3.3 Established trading processes compared to DLT trading processes	50
3.2.3.4 Common Ethereum smart contract standards.....	51
3.2.3.5 Structure of an Ethereum transaction.....	58

3.2.4	Gap analysis with respect to RTS 22.....	71
3.2.4.1	Fields similar to RTS 22.....	72
3.2.4.2	Fields not covered in RTS 22.....	73
3.2.4.3	Fields of particular importance.....	73
3.2.4.4	Fields relevant for on-chain analysis.....	77
3.2.5	Conclusion.....	78
3.3	Hyperledger Fabric.....	79
3.3.1	Background.....	79
3.3.2	Applied methodology.....	83
3.3.3	Main elements of a transaction.....	83
3.3.3.1	Definition of a Hyperledger Fabric transaction.....	83
3.3.3.3	Established trading processes and DLT trading processes.....	86
3.3.3.4	Structure of a Hyperledger Fabric transaction.....	86
3.3.4	Gap analysis with respect to RTS 22.....	94
3.3.4.1	Fields similar to RTS 22.....	94
3.3.4.2	Fields not covered in RTS 22.....	95
3.3.4.3	Fields of particular importance.....	95
3.3.4.4	Fields relevant for on-chain analysis.....	96
3.3.5	Conclusion.....	99
4	Annexes.....	100
4.1	Annex I.....	100
4.2	Annex II.....	102

Acronyms used

API	Application Programming Interface
CA	Certificate Authority/Certification Authority
CCP	Central Counterparty
CPU	Central Processing Unit
CSD	Central Securities Depository
DBMS	Database Management System
DLT	Distributed Ledger Technology
DLTR	Regulation (EU) 2022/858 of the European Parliament and of the Council of 30 May 2022 on a pilot regime for market infrastructures based on distributed ledger technology, and amending Regulations (EU) No 600/2014 and (EU) No 909/2014 and Directive 2014/65/EU
DLT MTF	DLT Multilateral Trading Facility
DLT SS	DLT Settlement System
DLT TSS	DLT Trading and Settlement System
DTI	Digital Token Identifier
DTIF	Digital Token Identifier Foundation
EC	European Commission
EIP	Ethereum Improvement Proposal
EOA	Externally Owned Account
EOL	End of Life
ERC	Ethereum Request for Comment
ESMA	European Securities and Markets Authority

ETH	Ether
EU	European Union
EVM	Ethereum Virtual Machine
FX	Foreign Exchange
GB	Gigabyte
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
ISIN	International Security Identification Number
ISO	International Organization for Standardization
KYC	Know Your Customer
MiFID II	Directive 2014/65/EU of the European Parliament and the Council of 15 May 2014 on markets in financial instruments and amending Directive 2002/92/EC and Directive 2011/61/EU
MiFIR	Regulation (EU) No 600/2014 of the European Parliament and of the Council on markets in financial instruments and amending Regulation (EU) No 648/2012
MSP	Membership Service Provider
MTF	Multilateral Trading Facility
NFT	Non-fungible Token
NCA	National Competent Authority
PoC	Proof-of-concept
PoS	Proof-of-stake
PoW	Proof-of-work
RTA	Registered Transfer Agent

RTS	Regulatory Technical Standard
RTS 22	Commission Delegated Regulation (EU) 2017/590 of 28 July 2016 supplementing Regulation (EU) No 600/2014 of the European Parliament and of the Council with regard to regulatory technical standards for the reporting of transactions to competent authorities
SDK	Software Development Kit
SEC	United States Securities and Exchange Commission
SHA	Secure Hash Algorithm
SS	Settlement System
TVTIC	Trading Venue Transaction Identification Code
UCITS	Undertakings for Collective Investments in Transferable Securities
URI	Uniform Resource Identifier
UTC	Coordinated Universal Time
XML	Extensible Markup Language

1 Executive Summary

Reasons for publication

The DLT Pilot Regime (DLTR) entered into force on 23 June 2022 and aims to foster innovation in the European Union's capital markets sector. It allows eligible firms to operate DLT market infrastructures to be used for trading and settlement purposes. A survey conducted during the ESMA workshop on the DLTR on 31 March 2022, identified three main DLTs (Corda, Ethereum, and Hyperledger Fabric) that might be used by DLT market infrastructures. The three DLTs are analysed in this study with respect to transaction reporting.

Transaction reporting plays a crucial role in current financial markets as it provides regulators with insights into market movements and trends as well as overall market stability. The objective of this study is to understand the implications of the use of DLT/blockchain in the context of transactions in financial instruments when an exemption to Article 26 of MiFIR is granted to a DLT market infrastructure. To do so, DLT developers, potential DLTR applicants from various European jurisdictions, and other stakeholders were interviewed to gain a better understanding of applicable data storage approaches and, more generally, transaction data produced by the DLT transactions. This practical knowledge was supplemented with theoretical knowledge gained by reading and analysing the DLTs' respective official documentation.

This document has been prepared for the European Securities and Markets Authority (ESMA) by PwC EU Services EESV (PwC). It reflects the views only of its authors, and the European Securities and Markets Authority is not liable for any consequence stemming from the reuse of this publication.

Contents

Sections 3.1, 3.2, and 3.3 take a closer look at the three DLTs analysed. In doing so, necessary background regarding the DLTs is outlined, considering certain specificities and appropriate use cases. Further, the sections dive into what constitutes a transaction on a DLT and bridges the gap to the definition of a transaction under Article 2, Paragraphs 2(a) and 3(a) of the RTS 22. The sections also explore the native fields produced as part of a DLT transaction, based on which a gap analysis is conducted. The gap analysis is made between the natively present fields of a DLT transaction and those fields to be reported under Article 26 of MiFIR. Lastly, suggestions are made regarding DLT transaction fields

that could be of relevance to a regulator and are not yet captured by the RTS 22 transaction reporting schema.

The general conclusion of the study is that there is a very limited number of transaction fields natively defined by the DLTs. This leads to significant gaps between the DLTs' transaction fields and the fields currently to be reported under the RTS 22 transaction reporting schema. Furthermore, there are also marked differences between the number of natively present and standardised transaction fields between the three DLTs.

At this point, it is recommended to extend the current RTS 22 transaction reporting schema by some of the DLT transaction fields deemed relevant. Especially important are the fields uniquely identifying DLT transactions, and, where applicable, those identifying the trading parties. Along these native DLT transaction the fields, the addition of the DTI is recommended, while it is also sensible to report on the smart contract address of an Ethereum transaction.

Generally, each of the technologies also allows to convey additional data, which could be included alongside a DLT transaction. Therefore, proper guidelines, potentially after gaining insights into market practices, need to be established to harmonise and standardise transaction reporting for the purpose of the DLTR and ensure market integrity and stability. All in all, the three DLTs do not fulfil the regulatory requirements under the RTS 22 transaction reporting schema out of the box. This makes it crucial that the DLT market infrastructures report all the relevant information required by regulators.

2 Introduction to the DLT Pilot and MiFID II/MiFIR transaction reporting

1. On 23 June 2022, the Regulation (EU) 2022/858 on a pilot regime for market infrastructures based on distributed ledger technology (DLT) (“the DLT Pilot Regime”, “DLTR”) entered into force. As part of the Digital Finance Package of the European Commission (EC), it furthers innovation and competition in the capital markets sector as it allows eligible firms to operate DLT market infrastructures.
2. Three types of DLT market infrastructures exist as part of the DLT Pilot. These are DLT MTFs, DLT SSs, and DLT TSSs. A DLT MTF is defined as a multilateral trading facility (MTF) which only admits to trading DLT financial instruments. A DLT SS, on the other hand, is a settlement system (SS) only settling transactions in DLT financial instruments against payment or delivery. A DLT TSS combines the services performed by a DLT MTF and a DLT SS.¹ To operate a DLT market infrastructure, firms must apply with their National Competent Authorities (NCAs).
3. The focus of this study is on the trading side of DLT MTFs and DLT TSSs, which will be referred to as DLT market infrastructures for the purposes of this report. Among other conditions, they are subject to the requirements that apply to multilateral trading facilities (MTFs) under Regulation (EU) 2014/600 (MiFIR) and Directive (EU) 2014/65 (MiFID II). However, the NCA can exempt DLT market infrastructures from some of said requirements if they comply with the conditions listed in Article 4 of the DLTR.²
4. Specifically, DLT market infrastructures may permit natural and legal persons to deal on own account via their systems, granted they fulfil a variety of requirements. In that case, additional measures to protect such participants may be required. Additional measures ought to be proportionate to the participants’ risk profiles. Moreover, DLT market infrastructures may be granted exemptions from transaction reporting requirements under Article 26 MiFIR. Should this exemption be granted, the DLT market infrastructures must nevertheless keep records of all transactions executed and further ensure that NCAs entitled to receive said data have direct and immediate access to it.
5. Initially set out for a duration of three years, the DLT Pilot will enable the trading of DLT financial instruments on DLT market infrastructures in the European Union, with the aim to stimulate innovation in the sector while guaranteeing investor protection. Annually, ESMA shall publish interim reports providing information on the functioning of the markets but also to provide clarifications on the Regulation’s application. Following the initial three-year

¹ <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32022R0858#d1e717-1-1>

² <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32022R0858#d1e1048-1-1>

period, ESMA will present a report to the European Commission covering, among other things, the number of DLT market infrastructures, an overview of DLT financial instruments admitted to trading and recorded, as well as an overall assessment of the DLT Pilot's costs and benefits. Furthermore, a recommendation as to whether and how the regime will be continued ought to be made.³

6. Over the course of its initial three-year period, the DLT Pilot imposes certain restrictions, for instance, on the financial instruments it covers. More precisely, it encompasses shares, bonds, and UCITS, which are subject to further thresholds pertaining to, among other things, the issuer's market capitalisation. Operators of DLT market infrastructures shall activate their respective transition strategies should the aggregate market value of all DLT financial instruments admitted to trading or recorded on that infrastructure exceed EUR 9 billion.⁴
7. MiFID II/MiFIR and their legal framework aim to protect investors in financial markets, while providing market transparency and functioning as a harmonised set of financial regulation in the European Union (EU). Overall, 28 Regulatory Technical Standards (RTS) are in place ranging from organisational requirements to disclosure obligations and the reporting of transactions.⁵ More precisely, Delegated Regulation (EU) 2017/590 (RTS 22) prescribe how transactions are ought to be reported in a consistent and standardised format to NCAs to enable said NCAs to be able to analyse the reported data effectively.
8. Article 2, Paragraphs 2(a) and 3(a) of the RTS 22 respectively define a transaction as the conclusion of an acquisition or disposal of a financial instrument.⁶ Such RTS 22 transactions must be reported no later than the close of the following working day and entail complete and accurate details regarding the nature of the financial instruments acquired or disposed of. In total, the current RTS 22 transaction reporting logic consists of 65 fields inquiring about information regarding the buyer and seller of a financial instrument, its details and transmission, as well as specific transaction details including quantity, trading date time, and price.⁷

³ <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32022R0858#d1e2493-1-1>

⁴ <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32022R0858#d1e966-1-1>

⁵ https://ec.europa.eu/finance/securities/docs/isd/mifid/its-rts-overview-table_en.pdf

⁶ https://ec.europa.eu/finance/securities/docs/isd/mifid/rts/160728-rts-22_en.pdf

⁷ https://ec.europa.eu/finance/securities/docs/isd/mifid/rts/160728-rts-22-annex_en.pdf

3 Registration of transactions on selected Distributed Ledger Technologies

9. The purpose of this study to gain some understanding of the specificities of the DLT to enable ESMA to have informed discussions on the topic in the context of subsequent market consultations on a broader set of DLTs. The three selected DLTs were considered as a good starting point for the analysis to ensure that the study could be delivered within the required timeframes.
10. Three DLTs, Corda, Ethereum, and Hyperledger Fabric, were selected to be analysed as part of this study. These DLTs were solely identified based on a survey launched during the ESMA workshop on the DLT Pilot Regime held on 31 March 2022. The participants in the workshop were individuals, or firms' representatives, who had responded to the ESMA Call for Evidence regarding the DLT Pilot Regime.⁸ Neither ESMA nor PwC are endorsing any of these DLTs or the softwares used to perform the study.

3.1 Corda

3.1.1 Background

11. Corda is a permissioned, peer-to-peer DLT by technology provider R3 HoldCo LLC ("R3").⁹ It aims to optimise existing processes in regulated markets and has hence found widespread application in the financial markets sector. Due to its setup as a permissioned DLT, information is shared between parties on a need-to-know basis. This means uninvolved third parties generally cannot access the stored information as there is no global broadcast of all Corda transactions. This allows for increased data privacy and Corda transaction privacy with involved parties being able to flexibly structure Corda transactions and deciding which data they intend to make available to third parties.
12. Peers on Corda are best defined as network nodes owned and operated by specific parties that have unique identities. On the Corda DLT, nodes are typically operated by legal persons rather than natural persons. This is due to the costs and complexities associated with node setup and the nodes' continuous operation. Nodes validate Corda transactions, and each maintain their own copy of the ledger, ensuring the DLT's security and integrity. Peers can communicate and enter into Corda transactions with another, essentially making the peers the network's participants. These activities are enabled in part by making use of

⁸ <https://www.esma.europa.eu/press-news/consultations/call-evidence-dlt-pilot-regime>

⁹ <https://r3.com/products/corda/>

X.509 certificates¹⁰. The X.509 certificate format was developed by the Internet Engineering Task Force (IETF) in 1988 and finds widespread application in networking and security protocols.¹¹

13. To keep the Corda network secure, X.509v3 certificates are issued to the peers, i.e., the network participants, upon joining the network. This is done by a network component called certification authority (CA), which is typically operated by a trusted entity. The CA could be the initiator of the network or an organisation that possesses knowledge on how to technically operate network infrastructures. CAs occupy a crucial role in maintaining Corda's security as they help ensure that network participants are correctly identified and thereby facilitate, for instance, the sending of Corda transactions. The certificates themselves then include, among other things, information on participants' public keys or their names.
14. States are further essential components of the Corda DLT. They are best defined as immutable objects stored on Corda's ledger and representing shared facts between network participants. A state's immutability ensures that it can neither be modified nor deleted once it is created.¹²
15. When such a fact changes, however, a new state is created by one of the network participants. To do so, the previous state serves as an input. Using a state as an input in a new Corda transaction is also called "consuming" a state. Once a state is consumed, it is marked as historic. Hence, an input state can be used to generate one or more new output states. This concept is explored further in Sections 3.1.3.1 and 3.1.3.2.
16. Current as well as historic states are stored in so-called vaults. Vaults are maintained by the network nodes. Every network node will thus have stored all applicable states, consumed and unconsumed, for the Corda transactions in which the node has been a participant.¹³ Hence, Corda does not have a central ledger recording facts for all network nodes but rather individual nodes storing the data known to them.
17. Corda further utilises so-called smart contracts, which digitise and enforce agreements entered into between various network participants. Smart contracts can be designed and implemented in a variety of ways to reflect and support the use cases they are applied to. Typically, they are implemented to put certain constraints on how states will evolve over

¹⁰ X.509 is a commonly used standard for public key infrastructure. X.509 certificates are used to bind an identity to a digitally signed public key. Among other things, they further contain information as to the certificate's issuer and its validity period.

¹¹ <https://sectigo.com/resource-library/what-is-x509-certificate>

¹² <https://docs.r3.com/en/platform/corda/4.8/enterprise/key-concepts-states.html>

¹³ <https://docs.r3.com/en/images/vault-simple.png>

the course of their lifetime. For instance, it can be specified that the coupon payment on a bond must remain at 2% of its notional amount over the course of its lifetime.

18. A further example is the transfer of a DLT financial instrument in exchange for e-money tokens between Party A and Party B. In such a scenario, the smart contract checks the balances of the two contracting parties to ensure Party A possesses the DLT financial instrument and Party B possesses enough e-money tokens to purchase it. This can be done by the smart contract querying the vault and assessing the respective balances of the parties or by checking the history of all Corda transactions the parties have been involved in. The correctness of this data is ensured, for instance, through the technical notarisation of Corda transactions the parties have been involved in. Upon the success of the check, the smart contract automatically executes the trade by transferring both the DLT financial instrument and the e-money tokens to their new owners, which are Party B and Party A respectively.
19. Another essential Corda component are the so-called flows. Corda provides a set of built-in flows that assist in automating tasks that commonly occur on the network. Flows can automate processes related to initiation, verification, or notarisation of a Corda transaction. More generally, flows enable inter-node communication. By facilitating communication between network participants, flows play a decisive role in coming to agreements regarding ledger updates.
20. In the financial sector, the Corda DLT has been used for a variety of activities, such as certain kinds of bond swaps or inter-custodian swaps between large financial institutions.¹⁴
21. This report takes into consideration Corda's Open Source Version 4.8.

3.1.2 Applied methodology

22. The methodology applied to analyse Corda transaction details and potential reporting mechanisms on Corda followed a two-fold approach. Firstly, a field study was conducted simulating a real-life Corda transaction to assess and verify natively implemented and produced Corda transaction fields. Test outputs were further analysed to gain a better understanding of the flexibility and use cases the technology covers. Secondly, R3 was approached to learn more about possible Corda transaction registration and data storage approaches.

¹⁴ Bank consortium Fnality International and Luxembourg company HQLAx successfully completed the first proof of concept (PoC) of a cross-chain repo swap settlement between Enterprise Ethereum and Corda.

23. More specifically, as a first step and as described in Figure 24, a test network was set up. This test environment was used to conduct a simple sample Corda transaction. The Corda transaction was subsequently analysed as to the information contained within it and provided the base layer of information, which will always be included in any Corda transaction.
24. As a second step, R3 was also approached to gain more insights into applicable use cases of the DLT. R3 mentioned that Corda is a network allowing various kinds of applications to be built on top of it. While the technology is especially suitable for settlement, trading applications also can be and are built on Corda. On the topic of trading, it was further said that the DLT is especially suitable for the trading of illiquid assets.

3.1.3 Main elements of a Corda transaction

3.1.3.1 Definition of a Corda transaction

25. Corda transactions are mechanisms by which states, as referred to in Paragraph 14, evolve over time. Generally, three types of changes can be facilitated by Corda transactions. Said types are issuances, i.e., the issuing of new states on the ledger, updates, i.e., the changing of properties of an existing state on the ledger, and exits, i.e., the consumption of a state on the ledger without the creation of a new one.
26. Consuming a state on Corda essentially means to utilise it as an input to a subsequent Corda transaction. As illustrated further in Figure 3, State₀ (zero) is consumed to create State₁ (one). State₁ is now the most up-to-date state and can be used for further Corda transactions.
27. The above definition of a Corda transaction poses a distinct difference to how transactions are defined within Article 2, Paragraphs 2(a) and 3(a) of the RTS 22.¹⁵ For the purpose of the DLT Pilot Regime, RTS 22 transactions as defined in Paragraph 8 of this report are in scope, rather than the more granular Corda definition.
28. Corda transactions are considered proposals, i.e., requests requiring verification through both parties signing with their respective X.509 certificates to update and evolve the ledger. Such proposals can be made by any network participants and contain the details of the proposed Corda transaction. Corda transactions may also be understood as sets of operations taking zero or more input states as reference and producing zero or more output states as a result.

¹⁵ https://ec.europa.eu/finance/securities/docs/isd/mifid/rtts/160728-rtts-22_en.pdf

29. As defined in more detail in Paragraph 47, a Corda transaction taking zero input states as reference would be an *issuance* transaction. An *issuance* transaction on Corda could be used to issue a bond or another kind of DLT financial instrument.
30. A Corda transaction consuming one or more input states without creating any output states, on the other hand, is called an *exit* transaction. An applicable example for an *exit* transaction is the maturity of a bond.
31. Corda transactions making use of one or more input states and producing one or more output states are called *update* transactions. An *update* transaction making use of one input state and creating one output state can be the sale of a bond from Party A to Party B. When selling the bond, the input state referenced would specify, among other things, the current owner of a bond. In this example, that is Party A. The output state, on the other hand, would specify the bond's new owner, hence Party B.
32. Once an output state is created, the input state is consumed, thereby marking it as historic. This means, once the bond has been transferred to its new owner Party B, the input state, which previously specified that Party A owned the bond, is now considered outdated. The newly created output state then serves as an input state for further, subsequent Corda transactions, such as a sale of the bond from Party B to Party C. Therefore, the current state of the ledger will contain all output states created in Corda transactions as long as they have not been classified as historic and been used in subsequent Corda transactions.
33. A further crucial component pertaining to Corda transactions is achieving consensus. For a Corda transaction to reach consensus and be technically accepted¹⁶ by the network, two criteria must be fulfilled: Uniqueness and validity.
34. A Corda transaction's technical validity can be asserted by examining whether the smart contract associated with the Corda transaction runs successfully and whether all required signatures, i.e., the signatures of the involved network participants, are present.
35. Transaction uniqueness on Corda, on the other hand, is given if no other Corda transaction has already consumed the same input state. Ensuring the uniqueness of a Corda transaction is crucial to avoid double-spending. In a practical example, this means if a Corda transaction entails the sale of a bond from Party A to Party B, the Corda transaction is unique if the bond in question has not yet been consumed.
36. However, should Party A already have sold its entire holdings of the same bond to Party C, then the bond has already been consumed and Party A can no longer sell it to Party B.

¹⁶ Technical acceptance means the Corda transaction fulfils the Corda network's predefined technical standards and boundaries.

This is because the consumption of a state leads to the state as being marked as historic and unable to be used in a further Corda transaction.

37. This could be likened to a more simplistic real-world example outside of Corda. For instance, Party A has already given Party C 100 EUR. Party A cannot then give the same 100 EUR to Party B, because they have already been spent, i.e., consumed, and the fact that Party A has the 100 EUR in its possession is no longer the most up-to-date state of facts.
38. Corda transaction validity can be ensured and agreed upon independently by the involved parties, often with the help of commands indicating the Corda transaction's intent. In other words, commands make a Corda transaction more human-readable. For instance, a command "couponPayment" may accompany the update of a bond on Corda. That way, it is clarified that there has been an update to the bond due to the coupon payment to the bond owner (see Figure 1 below for illustration purposes).
39. Another example is Party A selling a DLT financial instrument to Party B. Such a Corda transaction may be accompanied by a command "transferOwnership" to clarify that its intent is the transferral of ownership of the exchanged DLT financial instrument. Commands thus help make sense of a Corda transaction.

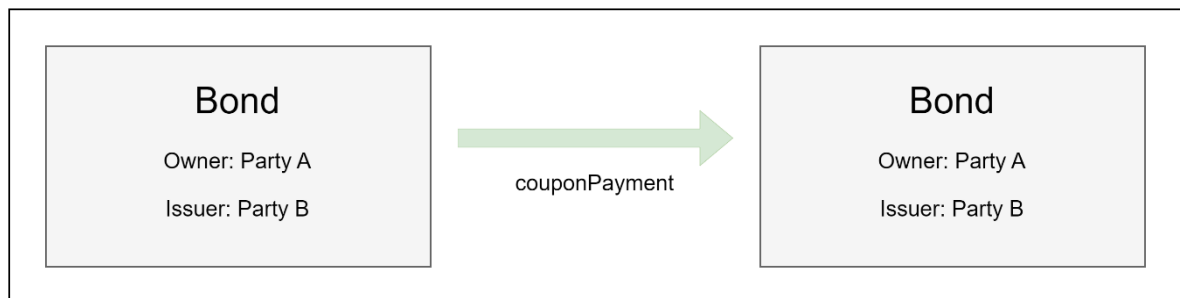


FIGURE 1: COMMAND INDICATING AN UPDATE TRANSACTION'S INTENT

40. Reaching consensus regarding a Corda transaction's uniqueness requires a predetermined, often independent, observer. As said observers are usually merely interested in the uniqueness of a Corda transaction, they typically do not see the full scale of the data for the to-be-assessed Corda transaction but rather focus on the consumed input state(s).
41. This task is performed by notary clusters, who must be the appointed notary cluster of all input states in connection with a Corda transaction and who, by signing it, provide a point of finality from a system's point of view. Corda transactions are committed to the ledger

once signed by the involved parties and validated by the notary. This means, the outputs generated in the Corda transaction have now become part of the ledger's current state.

42. Corda transactions can further be supplemented by various other components, assisting in the assessment of their uniqueness and validity. For instance, Corda transactions may be subject to time-windows specifying the period during which they can be committed. Time-windows are a useful tool for coordinating between the network participants involved while further preventing the submission of invalid and outdated Corda transactions. In cases, where a predetermined time-window has been set, the notary clusters mentioned above will intervene and refuse to commit Corda transactions outside of it, upholding network integrity.
43. Additionally, Corda transactions may contain attachments coming in the form of referenced .zip/.jar files. However, attachments are not mandatory and are referenced to by making use of a hash. This attachment hash is included within the Corda transaction and computed based on the attachment's contents. Further, the attachment hash is different from the hash of the Corda transaction it is applied to.
44. Attachments are mainly used for making data of interest available on the ledger. For instance, they can contain contract code or transaction metadata, such as relevant pricing data. For transaction reporting, an attachment could also include certain RTS 22 fields regarding the Corda transaction. However, as discussed in 3.1.4 and following, not all RTS 22 fields apply to Corda transactions.
45. The number or file size of attachments in a Corda transaction is not subject to any inherent limitations. However, each Corda network has a set of parameters that all nodes participating in it need to agree on and use to interoperate with one another correctly. Said parameters specify, for instance, the maximum allowed size of bytes contained in a Corda transaction, including its attachments.
46. Any type of file can be attached to a Corda transaction if it can be placed inside a .zip or .jar file.¹⁷ Therefore, the attachments may contain any type of data, including but not limited to documents, images, or even reports. As mentioned in Paragraph 44, it would also be possible to attach relevant RTS 22 information regarding a Corda transaction in XML format.

¹⁷ <https://training.corda.net/corda-advanced-concepts/attachments/>

3.1.3.2 Types of Corda transactions

47. *Issuances* stand at the beginning of a state’s lifecycle as an *issuance* creates new states without making use of or containing any input states (see issuance of state S_0 in Figure 2 below).¹⁸ Without utilising any input states, they can be used to, for instance, issue DLT financial instruments or other kinds of tokens on the ledger.¹⁹ Issuances typically do not necessarily need to be notarised as no states are consumed and, thus, no double spending can occur.
48. From a network perspective, there are natively no restrictions as to who can issue a new state on Corda. Theoretically, any Corda network participant could issue a new state on the network. As Corda is a permissioned technology, however, this access could also be restricted to only allow selected participants to issue states. In the context of financial markets, a network could be restricted to only allow certain banks or other financial institutions, such as DLT market infrastructures, to issue new states, i.e., new DLT financial instruments, on Corda. The decision of who can issue new states on a network is made by the entity writing the application, i.e., by the DLT market infrastructure operating a Corda network.

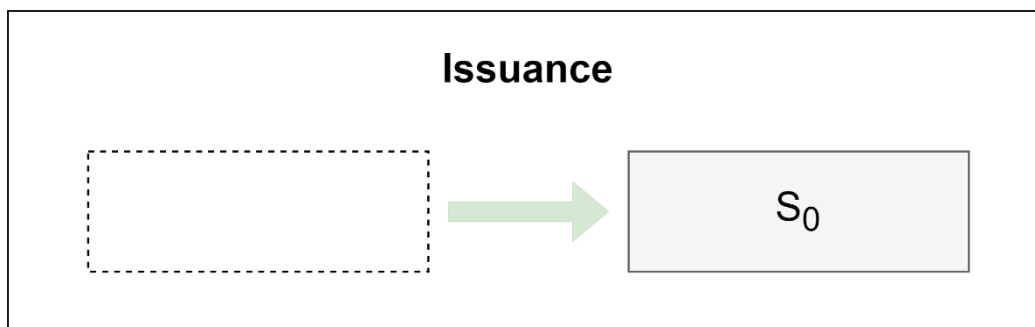


FIGURE 2: ISSUANCE OF A STATE ON CORDA

49. *Updates*, unlike *issuances*, consume one or more input states to produce one or more output states (see update of state S_0 to state S_1 in Figure 3 below). The necessary information regarding the input state is retrieved by querying the vault. A viable example is

¹⁸ Per the current understanding of the DLT Pilot Regime, financial instruments already existing off-chain, i.e., not on a DLT, can also be re-issued on a DLT. Similarly, the Pilot also allows for the simultaneous offering of a financial instrument off-chain and on-chain, i.e., on a DLT. It should be noted that this may open up opportunities for arbitrage in cases where no link between the off- and on-chain financial instruments exist. Arbitrage possibilities could potentially be minimised through the integration of oracles. The concept of oracles is further explained in Paragraph 74.

¹⁹ In the context of financial instruments, Corda allows for the creation of so-called “digital twins”. This means, financial instruments that already exist outside of the DLT, i.e., shares of a publicly traded company, can be re-issued and made tradable on the DLT. This may, however, pose further legal and regulatory questions. For example, it would give rise to the question of whether a digital twin would also be considered a financial instrument, even in cases where it is only used for settlement purposes.

Party A selling a portion of a bond to Party B in exchange for cash. In such a case, the vault would contain information with respect to the overall bond value held by Party A and the amount of cash held by Party B.

50. Corda transactions happen atomically to prevent the risk of one party ending up with both the cash to be transferred and the bond to be sold. This means that, within the Corda transaction, all state changes must occur and be successful, or the transaction is deemed to not have happened. This means that both the transfer of the bond portion from Party A to Party B as well as the transfer of cash from Party B to Party A must be successful for the entire Corda transaction to be successful. Due to this implementation, hence, monitoring and confirmation mechanisms are made obsolete.

51. *Updates* on Corda can also have an uneven number of input and output states. An example of a Corda *update* transaction with fewer inputs than outputs would be the sale of a bond, which has accrued interest. If Party A, which is in possession of a bond making a coupon payment every 90 days, sells the bond to Party B 30 days after the last coupon payment, Party A would receive two outputs, respectively for the value of the bond and the value of the accrued coupon.

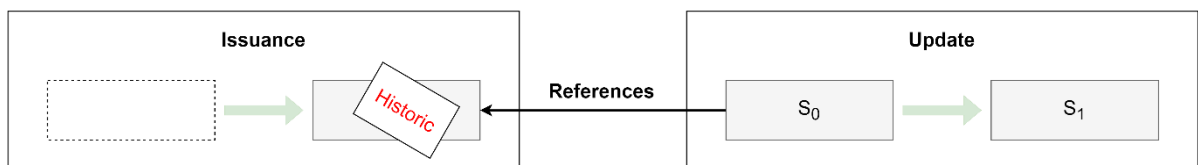


FIGURE 3: UPDATE OF A STATE ON CORDA

52. Exits come at the end of a state's lifecycle as they end the state chain by utilising one or more input states without creating any further output states (see exit of state S_1 in Figure 4 below). As a node's vault on Corda only contains active states, successfully exited states will not be part of such a vault and cannot be queried from it any longer. However, the state's trail will remain available, and the state's historical versions can continue to be accessed and audited. Exits occur, for instance, when a bond reaches maturity and no longer needs to be displayed on the ledger.

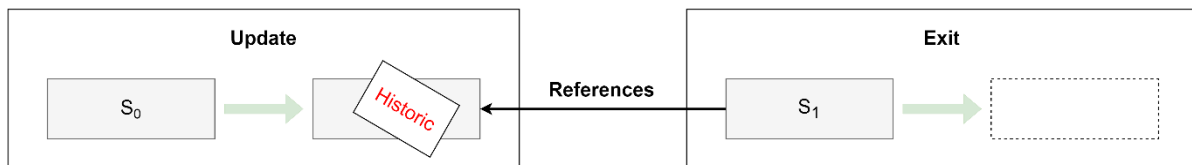


FIGURE 4: EXIT OF A STATE ON CORDA

53. As Corda is a peer-to-peer network using so-called point-to-point messaging, flows are used to facilitate the communication between nodes and simplify the process of entering into agreements to update the ledger, i.e., entering Corda transactions. Because multiple steps are involved when attempting to enter a Corda transaction, including but not limited to the initial creation of the proposal, its signing and sending, flows provide a way to automate this process.
54. Flows can thus be understood as the building blocks to construct Corda transactions. For instance, when one node on Corda wants to send a Corda transaction to another node, it can initiate a flow guiding the other node through verifying and committing the Corda transaction to its local ledger. Generally, flows are designed to be flexible and modular for them to be easily customised and combined to support a variety of interactions and Corda transactions.
55. As DLT market infrastructures are part of the DLTR and help facilitate transactions in DLT financial instruments, they can also assess whether these transactions are executed successfully. In a case, where Party A and Party B want to enter a Corda transaction and exchange DLT financial instruments for e-money tokens, the DLT market infrastructure facilitating this action can see whether the two parties actually exchanged ownership of the DLT financial instruments and e-money tokens respectively. To do so, the DLT market infrastructure could require the contracting parties to query their respective vaults and provide the queried data. When reading out the data, it will become apparent whether the balances of the two parties have changed accordingly or remained the same.
56. Furthermore, Corda’s point-to-point messaging differs from global broadcast models insofar as that Corda transactions are never received by any peers uninvolved in them. Rather, peers must specify the recipients of the messages they send. Should a regulator want to be involved and obtain information regarding the Corda transactions between peers, various ways to enable this can be thought of.
57. In the “Report on the DLT Pilot Regime – Study on the extraction of transaction data”, the topic of extracting information regarding Corda transactions in DLT financial instruments by an external party, such as a regulator, is discussed in more detail. Further, it would be

possible for a regulator to participate in the defined Corda transaction flows and thereby directly receive reports regarding the conducted Corda transactions.

58. Technically, there is no limit to how many actions may occur within one Corda transaction. Corda is designed to effectively handle large volumes of Corda transactions of different kinds of complexity. Its performance may nevertheless be affected by the file size of the Corda transactions to be handled.
59. Larger Corda transactions may bring with them an elongated validation process and thus take longer to be committed to the ledger. Hence, the network parameters mentioned in Paragraph 45 may be defined to restrict the maximum (or minimum) sizes allowed for Corda transactions to minimise performance issues. Such parameters, therefore, may contribute to a smooth and efficient functioning of the technology and the conducted Corda transactions.

3.1.3.3 Established trading processes compared to DLT trading processes

60. In the current financial markets, the trading and settlement process involves multiple steps and several intermediaries that stand between the buyer and seller. The current process takes place over the span of two days, from T+0, the day the two parties' respective bids and asks are matched to T+2, when the involved cash and securities are settled. An abstracted security trading and settlement process can be seen in Figure 5 below.

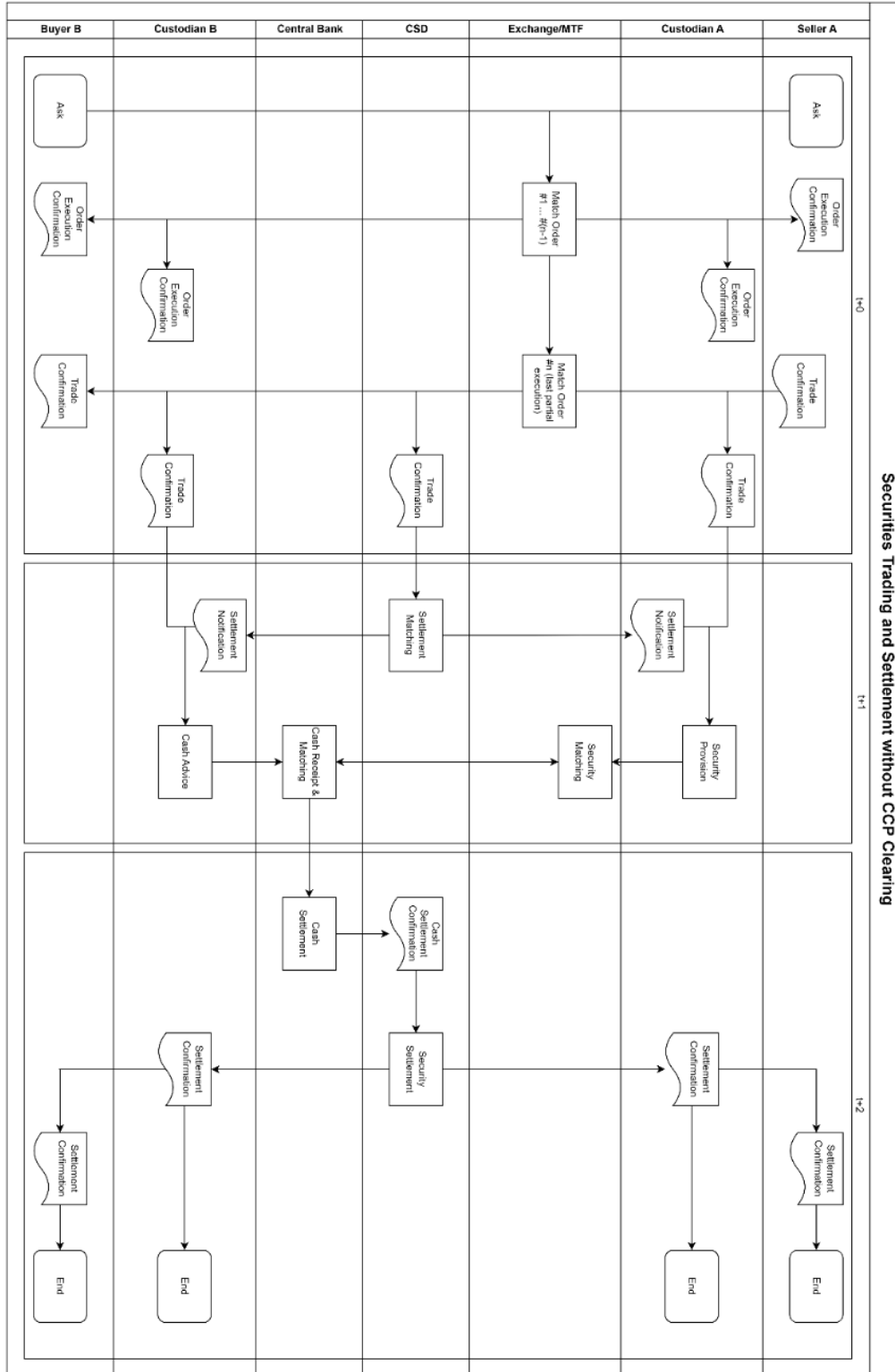


FIGURE 5: EXEMPLARY TRADITIONAL TRADING PROCESS

61. In the outlined scenario, the buyer's and seller's respective bid and ask are matched by an exchange, in this case an MTF. Upon matching the bid and ask, an order execution confirmation will be transmitted to the two trading parties as well as their respective custodians. Orders can also be partially executed. In such a case, the order execution confirmation process is repeated until the last partial execution occurs by matching the last remaining order. Once this happens, the MTF transmits trade confirmations to the two trading parties, both custodians, as well as a central securities depository (CSD). These processes typically happen on the day of the trade, thus on T+0.
62. On T+1, the day following the trade, settlement matching occurs via the CSD. The CSD also transmits a settlement notification to the custodians of the trading parties. The custodian of the seller then provides the security, whereas the buyer's custodian offers a cash advice. While the MTF matches the securities received from the seller's custodian, the cash advice is channelled through the central bank, where further matching occurs. This concludes the activities on this day before the trading and settlement process is finalised on T+2.
63. On T+2, cash settlement occurs through the central bank, which transmits a cash settlement confirmation to the CSD. The CSD then is responsible for the settling of the securities. Once the securities have been settled, settlement confirmations are transmitted to both custodians as well as both trading parties, i.e., the buyer and the seller. Once all settlement confirmations are transmitted, the trading and settlement process is finalised and ends.

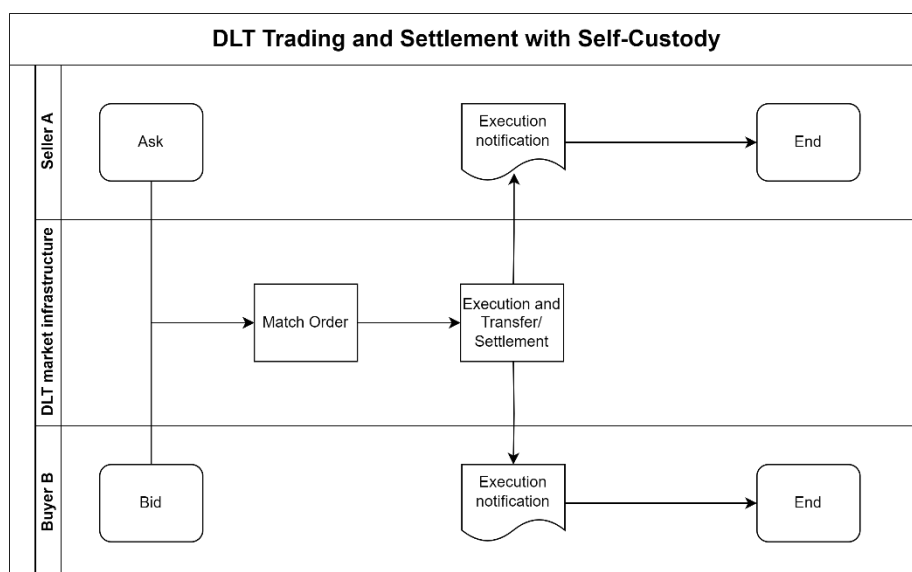


FIGURE 6: DLT SECURITY TRADING PROCESS WITH SELF-CUSTODY

64. As shown in Figure 6, a DLT trading and settlement process involving self-custody vastly reduces the number of involved parties, as self-custody negates the need for an external custodian. Also, it is assumed that a licenced and regulated e-money token has been issued on the network, thereby negating the need for a third-party payment provider. The entire process now only involves the two trading parties along with the DLT market infrastructure as the last remaining intermediary. The buyer's and seller's accounts, e.g., their wallets storing and managing their tokenised securities and e-money tokens, are connected to the DLT market infrastructure.
65. Again, the bid and ask are matched by the DLT market infrastructure, which initiates the settlement by executing the transfer of the tokenised security from the seller's connected account/wallet. To finalise the transaction, both trading parties will typically receive a notification that the trade was executed successfully and that their balances have been updated accordingly.
66. It is important to note that, for the purpose of the DLT Pilot Regime, Figure 6 may in some cases have to be extended. This is because the DLTR does not allow for payments to be conducted in cryptocurrencies, which may be natively present on a DLT. Instead, ISO currency and e-money tokens must be used to engage in DLT financial instrument transactions. In case such a registered and licenced e-money token is issued and therefore present on a DLT, Figure 6 holds true. If not, a third-party payment provider may have to be integrated for payment purposes, thereby adding another intermediary to Figure 6.

3.1.3.4 Structure of a Corda transaction

67. Corda transactions are characterised by their flexibility. Hence, their structure and the granularity of the information provided and extractable depend greatly upon the envisioned use cases. At the very least, a Corda transaction will contain information regarding its transaction hash (*txhash*) along with the applicable *index*.
68. Corda transactions also produce a timestamp (*recordedTime*) specifying the time at which a Corda transaction was committed to the ledger. Further details, such as contract or notary information, i.e., information as to the identity of the node notarising the transaction, will also be included in the Corda transaction but may not apply to the executed Corda transaction and thus be empty.
69. Furthermore, additional state data, for instance, regarding the Corda transaction's value or the involved parties, can be appended to the Corda transaction. However, this is not mandatory and, thus, not always the case. As mentioned in Paragraph 16, network nodes store information relevant to the respective Corda transactions they have been involved in within a vault. Vaults can be accessed via various queries that retrieve the information stored in the vault.

70. As discussed in Paragraphs 47 and 48, Corda allows for the issuing of new states. The issuance ability, however, can be restricted by the network operator, e.g., a DLT market infrastructure, to only allow certain eligible network participants to issue new states in a given Corda network. This allows, for instance, to restrict the issuance ability to selected network participants, such as financial institutions, DLT market infrastructures, or other competent entities. Doing so, would only allow these eligible entities to issue DLT financial instruments.
71. When a new state, or in this case a new DLT financial instrument, is issued, the applicable smart contract will be attached to it. The logic implemented in the smart contract then defines how this state, i.e., this DLT financial instrument, can evolve over the course of its lifetime. This process was briefly touched upon in Paragraph 17. The link between the state and its accompanying smart contract cannot be broken.
72. For instance, for a DLT financial instrument, it could be mandated that Corda transactions in the DLT financial instrument must always specify the price, at which the DLT financial instrument was acquired or disposed of, or the quantity of the DLT financial instrument acquired or disposed of. To do so, it could be specified in a smart contract that a Corda transaction, which does not contain this information, will never produce a valid Corda transaction.
73. Hence, Corda's inherent flexibility can be restricted in a way to allow a regulator to mandate the design of a smart contract attached to a financial instrument issued on the DLT for it to produce a valid Corda transaction in DLT financial instruments. There should be a comprehensive analysis of any advantages and disadvantages such a mandate would bring about.
74. Further, it is inherently not possible to include information within smart contracts that is not present on the ledger. Hence, oracles²⁰ may need to be used so data outside of the Corda DLT can be provided to smart contracts. Oracles are services finding application in distributed ledger technology to provide "real world" data within the DLT ecosystem.²¹
75. These services can be used for a variety of purposes, such as providing pricing or exchange rate information. Making use of such oracles would also allow for the integration of further potentially relevant information on Corda. The Digital Token Identifier Register (DTIF Register), for instance, carries additional information (see Paragraph 94) regarding the digital tokens listed on it which could be integrated via oracles.²² The integration of

²⁰ <https://training.corda.net/corda-advanced-concepts/oracles/>

²¹ For further reading on oracles: <https://learn.radixdl.com/article/what-is-an-oracle>

²² <https://dtif.org/registry-search/>

oracles, however, may lead to additional security measures having to be taken to mitigate associated risks.

76. As mentioned before, the DLTR covers the use case of DLT market infrastructures being involved and acting as intermediaries between the parties to a trade in DLT financial instruments. In theory, Corda network participants cannot be prevented from bypassing the DLT market infrastructure and transacting with one another directly by specifying their own Corda transaction flows.
77. However, the smart contracts attached to the DLT financial instruments tradable in a Corda network can be specified in such a way as to require a DLT financial instrument to be traded through a DLT market infrastructure. Hence, the bypassing of a DLT market infrastructure, and thereby peer-to-peer dark trading, can be prevented by specifying that a peer-to-peer trade can never produce a valid Corda transaction.
78. On the topic of Corda transaction identifiers, it should be noted that the below paragraphs outline various theoretical options to design a Corda transaction in DLT financial instruments on the Corda DLT. In doing so, it explores how Corda's flexibility could be applied to gain insight into Corda transactions in DLT financial instruments.
79. If a regulator intends to see certain specific information, and thereby a specific detail about a Corda transaction, the information in question needs to be included in the smart contract at the time the smart contract is deployed or be integrated in a transaction report another way. For instance, it could be enriched post-trade by linking it with information stored outside of the DLT. This depends on the design of the architecture of the DLT market infrastructure and potentially the kind of data in question.

a) Transaction identifiers

80. Corda transactions are uniquely identifiable by their *txhash* and *index*. Regardless of the applicable use case, both identifiers are always included in any Corda transaction. The DLT makes use of so-called Secure Hash Algorithms (SHAs), converting data into hashes containing 256 bits, better known as SHA-256 hashes. These hashes are attached to the Corda transaction and can be used to refer to a specific Corda transaction on the network. The *index* further references a specific output that is part of said Corda transaction.
81. **Example:** Party A sends 100 EUR to Party B in exchange for five shares of a company's stock. The Corda transaction, in which both the cash and the shares are exchanged atomically, will be assigned a unique SHA-256 *txhash*, for instance, "5cc2fba...". This hash is immutable and always refers to said Corda transaction between the two parties.

82. Within the Corda transaction itself, two updates occur simultaneously. However, these two updates do not constitute two separate Corda transactions. Rather, they are two activities occurring within a single Corda transaction. In this example, Party A sends the cash to Party B while, at the same time, Party B sends the shares to Party A. As part of the Corda transaction, both updates will be assigned a unique *index*, in this case “0” and “1” respectively. An *index* will thus reference a specific output within a given Corda transaction. This process is outlined in Figure 7 below.

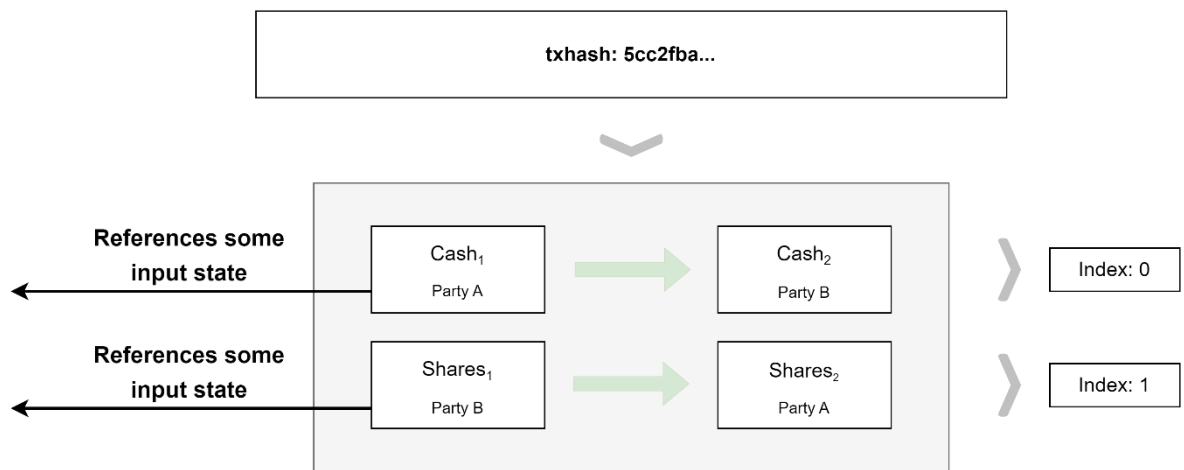


FIGURE 7: SAMPLE CORDA TRANSACTION BETWEEN TWO PARTIES

b) Party identifiers

83. Generally, network participants on Corda are identified by making use of X.509 certificates²³ mentioned in Paragraphs 12 – 13. It is required that the certificates used, follow the X.509v3 standard.²⁴ As mentioned in Paragraph 13, the X.509 certificate is issued by a trusted entity upon initially joining the network. On the Corda Enterprise²⁵ version, for instance, R3 themselves are in charge of issuing the certificate and managing the identity. In other cases and networks, a trusted and competent entity may be a DLT market infrastructure, a regulator, or a trade association.

84. Such certificates provide further details as to the identity of the parties engaging in a Corda transaction as well as their role within the network. Relevant details included within the

²³ For further reading regarding X.509 certificates: <https://learn.microsoft.com/en-gb/azure/iot-hub/reference-x509-certificates>

²⁴ <https://docs.r3.com/en/platform/corda/4.8/enterprise/network/permissioning.html>

²⁵ Corda’s Open Source and Enterprise version are functionally identical, whereas the Enterprise version offers additional, non-functional services. For further reading: <https://stackoverflow.com/questions/58493211/what-can-be-achieved-in-enterprise-corda-is-not-achievable-in-community-version#:~:text=Corda%20Open%20Source%20and%20Enterprise.24%20X%207%20Support%2C%20etc.>

certificate are, for instance, the subject to be identified, the issuer of the certificate and the issuer's signature, a serial number, and the subject's public key.

85. Anyone that is part of a Corda network, including trading venues, investment firms, regulators, etc. possesses an X.509 certificate for purposes of identification. The certificate itself can be explained as being stored within the respective nodes.²⁶ Hence, X.509 certificates provide a way of verifying and authenticating a node's identity. It may be sensible to require DLT market infrastructures to identify the operator of a Corda node in accordance with RTS 22 requirements as part of a Know Your Customer (KYC) process if the operator intends to engage in Corda transactions in DLT financial instruments. KYC processes are standard procedures implemented in the financial industry aimed at protecting financial institutions against fraud and other forms of wrongdoing.²⁷
86. Moreover, while Corda nodes are generally not operated by natural persons, nodes can host further accounts to be used by natural persons. This is achieved by a partitioning of a node's Corda vault into multiple subsets. These subsets then represent accounts, which can be used for customer or employee accounts, or any other account types.²⁸ Node operators themselves can decide whether they want to create accounts and what these accounts ought to be used for.
87. Although, accounts do not have a unique identity associated with them, they can still transact with accounts hosted by other nodes, accounts hosted by the same node, or even regular nodes themselves. Further, they can be identified in case they indeed engage in Corda transactions, for instance, through their *Account ID*, which should be unique at network level.²⁹
88. Hence, a financial institution identifiable by its extended X.509v3 certificate and operating a node on a Corda network could create multiple accounts for employees or customers. In theory, the users of the account could then engage in Corda transactions in DLT financial instruments and would need to be properly identified.
89. In conclusion, the DLT itself does not guarantee that the parties involved in a Corda transaction are identified in accordance with the current RTS 22 requirements. Therefore, it is important that there are regulatory requirements on the DLT market infrastructure to implement KYC-processes to ensure proper identification of trading parties.
90. **Example:** As outlined in Paragraph 83, any network node on Corda is identifiable by X.509v3 certificates. These certificates allow for extension and customisation by supporting

²⁶ <https://corda.net/blog/corda-firewall%E2%80%8A-%E2%80%8Acomponents-pki-deployment/>

²⁷ <https://www.swift.com/de/node/235031>

²⁸ <https://docs.r3.com/en/tools/accounts/accounts-index.html>

²⁹ <https://github.com/corda/accounts/blob/master/docs.md>

further arbitrary fields.³⁰ A possible approach to appropriately identify the parties to a Corda transaction would be to make use of these certificate extensions and customise them in such a way as that they would include further data of interest, such as personally identifiable data or RTS 22 relevant data.

91. In case the operator of a Corda node hosts further accounts to be used by natural persons for the purpose of engaging in Corda transactions, information regarding the natural persons should also be collected. This can be done by using a globally unique *Account ID*. This *Account ID* could then be linked with personal data regarding the natural person, which is collected during a KYC-process. A potential setup is displayed in Figure 10.

c) Object identifiers

92. Object identifiers are not natively provided for in Corda transactions. Thus, if object identifiers, such as an ISIN and DTI, are to be included in a Corda transaction, they must be defined and implemented in the respective smart contracts of the DLT financial instruments. However, as smart contracts on Corda and Corda transactions can be structured arbitrarily complex or rudimentary, not all of them must possess details as to the object included.
93. Furthermore, as mentioned in Paragraph 74, the link to the registers carrying ISINs and DTIs could provide for the provision of additional data regarding DLT financial instruments. Such a link could be established by making use of oracles in order to make this external data available within the network.
94. Generally, the integration of a DTI as an additional object identifier arguably brings with it multiple benefits. For instance, a DTI identifies the underlying DLT on which the DLT financial instrument is traded, while also providing information regarding the token's mechanism³¹ and its technical reference. The DTI thus provides for the technical attributes regarding a traded token. Furthermore, a DTI can also contain a link to the ISIN and therefore provides a one-to-many relationship if a DLT financial instrument is tradable on multiple DLTs. Overall, the DTI serves a double function as it further identifies both the DLT financial instrument as well as the DLT on which the DLT financial instrument is traded.
95. **Example:** The DLT Pilot Regime allows for both the issuing of entirely new DLT financial instruments as well as reissuing existing financial instruments in a DLT environment. Corda itself, however, does not natively provide for an applicable field in which this information can be stored. To nevertheless integrate an object identifier as part of a Corda transaction, one possibility is to include such an identifier, e.g., an ISIN or DTI, within the DLT financial

³⁰ <https://www.ibm.com/docs/en/external-auth-server/2.4.3?topic=securing-x509-extensions>

³¹ Per the DTIF, the token mechanism of an Auxiliary Digital Token is the "protocol used to create an auxiliary digital token".

instrument's smart contract. That way, the object identifier will be stored and always present on the network. Further components and characteristics of the DLT financial instrument could also be specified within the smart contract. Among those could be identifiers regarding the DLT financial instrument's issuer, its maturity data, or any other relevant data elements.

96. Figure 8 below shows how the contract state for a DLT financial instrument on Corda is defined. It includes such information as an ISIN and DTI to uniquely identify the DLT financial instrument. As outlined, further necessary details regarding the DLT financial instrument could also be specified.

```
import net.corda.core.contracts.Amount
import net.corda.core.contracts.ContractState
import net.corda.core.identity.AbstractParty
import net.corda.finance.contracts.asset.FungibleAsset
import java.util.*

// Acme Equity Token State
data class AcmeEquityTokenState(
    val isin: String,                // ISIN (International Securities Identification Number)
    val dti: String,                // DTI (Digital Token Identifier)
    val quantity: Amount<TokenType>, // Quantity of tokens
    val price: Amount<Currency>,    // Price per token in a specified currency
    val issuer: AbstractParty,      // Issuer party
    val owner: AbstractParty        // Current owner party
) : ContractState, FungibleAsset<TokenType> {
    override val participants: List<AbstractParty> get() = listOf(issuer, owner)
}

// Token type definition
data class TokenType(
    val identifier: String,         // Identifier for the token
    val fractionDigits: Int        // Number of decimal places for the token quantity

    // ... further necessary fields
)
```

FIGURE 8: HYPOTHETICAL DLT FINANCIAL INSTRUMENT CREATION

d) Price, quantity, and conversion mechanisms

97. Similar to the object identifiers mentioned above, information concerning price and quantity are neither mandatory nor standardised components to be included in a Corda transaction either. Therefore, the parties to a Corda transaction may decide to specify the number of units exchanged and their value but generally are not required to. Furthermore, the parties to a Corda transaction are free to choose the name and format of these fields.
98. Again, it could be mandated that any Corda transaction in DLT financial instruments must provide information regarding the price at which the instrument was acquired or disposed of. Similarly, information regarding the quantity of instruments acquired or disposed of could also be mandatory.

99. Prior to its actual execution, the parties to the Corda transaction must specify the quantity of DLT financial instruments they intend to acquire or dispose of. To do so, the instrument's smart contract must provide for a field, in which this information can be entered. A smart contract could be specified that, in a case where the quantity is not entered, the Corda transaction is invalid.
100. To ensure accurate pricing of DLT financial instruments, either an oracle would have to provide up-to-date pricing data regarding the instrument, e.g., if it already exists outside of the DLT, or the price of the instrument would have to be determined in another way, e.g., by supply and demand on the network itself. Another approach could be to simply specify a fixed price for an instrument.
101. The latter approach, however, would not be in accordance with the way a DLT market infrastructure is supposed to operate. DLT market infrastructures are subject to the same requirements as those applying to MTFs under Regulation (EU) No 600/2014 and Directive 2014/65/EU.³² Per definition, hence, they bring together third-party buying and selling interests.³³
102. It is important to mention that Corda does not have a native currency in which Corda transactions in DLT financial instruments can be executed. Hence, one possibility to exchange cash against a DLT financial instrument on Corda would be to integrate an external payment provider via an API. In such a case, a DLT financial instrument could be acquired or disposed of on the DLT but remain "locked", i.e., reserved, until the accompanying payment has been successfully sent. Once the payment has succeeded, the instrument could be "unlocked" and transferred to its new owner.
103. In the case where a DLT financial instrument is acquired or disposed of by exchanging it against e-money tokens, a conversion mechanism could be provided allowing the e-money tokens to be exchanged against standard ISO currency. For this approach, it may also be sensible to provide the conversion service via an external payment provider. Furthermore, if an e-money token is used for the acquiring of DLT financial instruments, it may too be sensible to include the e-money token's DTI for identification purposes in reports provided to regulators.
104. If the e-money token is solely issued on the DLT, one possibility to maintain its FX rate in relation to ISO currency would be to again make use of oracles. These oracles can feed information from the outside world into the DLT and ensure an accurate e-money token price.

³² <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32022R0858#d1e717-1-1>

³³ <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014L0065&from=de>

105. **Example:** As outlined, Corda does offer the option of specifying a price as part of a Corda transaction, however it is not a mandatory component. For Corda transactions in DLT financial instruments, various options for identifying the accompanying price and quantity can be envisioned. Examples of possible approaches are outlined in the following paragraphs.
106. According to the DLT Pilot regime, payments made for the acquisition of DLT financial instrument can be denominated in ISO currency or e-money tokens. To conduct such payments, a DLT market infrastructure operator could work with a third-party payment provider offering payment solutions in fiat currency or e-money tokens.
107. A further approach could be for DLT market infrastructure operators to issue their own e-money token which is to be used on their platform. In both cases, however, it should be noted that e-money tokens would have to undergo applicable licensing processes prior to being able to use them in the context of the DLT Pilot Regime.
108. As mentioned in Paragraph 105, the pricing of DLT financial instruments can be determined in different ways. For DLT financial instruments that already exist outside of the Corda DLT and were simply reissued on a DLT market infrastructure running on Corda, it may be sensible and appropriate to provide up-to-date pricing information by making use of oracles. For DLT financial instruments only available for trading on a specific Corda network, DLT market infrastructures would likely be tasked with determining the price of the instruments. This would occur as part of DLT market infrastructures' task of matching buying and selling interests.
109. Whether conversion mechanisms are necessary depends upon whether payments on a specific DLT market infrastructure are conducted in fiat currency or e-money tokens. In cases where payments are conducted in e-money tokens, it may be sensible for DLT market infrastructures to provide conversion mechanisms on their platform. Moreover, it may be sensible to also provide information on the exchange rate between the e-money token and its reference currency.
110. Figure 9 below outlines the hypothetical output of a Corda transaction, in which 100 units of Party A's 300 units of a fictitious DLT financial instrument (ISIN: FR1234567890, DTI: 71ABC1TK3) were transferred to Party B. The DLT financial instrument's price in this example is denominated in ISO currency and specified as 10 EUR. After the successful transfer of the DLT financial instrument, the trading parties' respective holdings of the DLT financial instrument are updated accordingly.


```

[INFO] 2023-05-19T12:00:00.000Z: Initiating flow for AcmeEquityTokenState transfer
[INFO] 2023-05-19T12:00:01.000Z: Starting AcmeEquityTokenState transfer flow
[INFO] 2023-05-19T12:00:02.000Z: Fetching AcmeEquityTokenState with ISIN FR1234567890 and DTI 71ABC1TK3
[INFO] 2023-05-19T12:00:03.000Z: AcmeEquityTokenState found. Quantity: 300 units.
[INFO] 2023-05-19T12:00:04.000Z: Transferring 100 units from A to B
[INFO] 2023-05-19T12:00:05.000Z: Creating new AcmeEquityTokenState for B
[INFO] 2023-05-19T12:00:06.000Z: New AcmeEquityTokenState created for B. Quantity: 100 units.
[INFO] 2023-05-19T12:00:07.000Z: Updating AcmeEquityTokenState for A
[INFO] 2023-05-19T12:00:08.000Z: AcmeEquityTokenState updated for A. Quantity: 200 units.
[INFO] 2023-05-19T12:00:09.000Z: Transaction completed successfully.

Transaction Hash: 7bc4fa...
Inputs:
- AcmeEquityTokenState [ISIN: FR1234567890, DTI: 71ABC1TK3, Quantity: 300, Price: 10 EUR, Issuer: BigBank, Owner: A]
Outputs:
- AcmeEquityTokenState [ISIN: FR1234567890, DTI: 71ABC1TK3, Quantity: 200, Price: 10 EUR, Issuer: BigBank, Owner: A]
- AcmeEquityTokenState [ISIN: FR1234567890, DTI: 71ABC1TK3, Quantity: 100, Price: 10 EUR, Issuer: BigBank, Owner: B]

Commands:
- CustomerMoveFungibleTokens [Token: AcmeEquityTokenState, Quantity: 100, Participants: A, B]

Signatures:
- A: abcdef1234567890
- B: 1234567890abcdef

```

FIGURE 9: HYPOTHETICAL CORDA TRANSACTION OUTPUT

e) Other attributes

111. Besides the always present information regarding a Corda transaction, which are the *txhash*, its *index*, and the *recordedTime* further information may also be present, depending on the design of the Corda transaction. In case the Corda transaction was notarised or contains a command indicating its intent, both these pieces of information will also be included in the Corda transaction. Besides the mentioned characteristics, Corda's flexibility allows for various other attributes or pieces of information that could be part of a Corda transaction. This is due the fact that Corda can handle and execute arbitrarily complex codes and smart contracts.
112. Complexity or richness of information pertaining to a Corda transaction in DLT financial instruments can be further enhanced by making use of oracles. As mentioned in Paragraph 74, oracles can be used to make real world information (information that is not present on the ledger) available on the ledger. This could range from additional reference data regarding the DLT financial instruments traded on Corda, to additional data regarding an issuer, or network participant or even information regarding noticeable real-time news events.
113. **Example:** A piece of information that could be included in a Corda transaction in addition to the fields mentioned above, is information regarding the purpose, i.e., the intent of the Corda transaction such as *Correction*. As mentioned in Paragraphs 38 and 39, Corda transactions can be supplemented by human-readable commands that specify why it is sent. The addition of such a field could make market and trade surveillance more transparent from a regulatory perspective.

114. Another example of a piece of information that could be included in a Corda transaction in DLT financial instruments would be, if applicable, the credit rating of the issuer of the instrument. In situations where a company, like a financial institution, has issued a bond that can be traded on Corda, it may be possible to set up a system where whenever the bond is bought or sold, the credit rating of the company that issued the bond is reported. Overall, different implementations are conceivable.

f) Storage of additional business fields

115. There are various ways in which additional business fields, i.e., such fields that are not natively a part of a Corda transaction, can be stored. As outlined over the course of this chapter, Corda possesses a limited amount of natively implemented fields that are standard components of any Corda transaction. Therefore, for the purpose of transaction reporting in accordance with RTS 22, an approach ought to be found that allows for the reporting of additional data beyond the native Corda data, in case such data is relevant and desired from a regulator's point of view.

116. One approach would be to store further relevant data on the ledger. In terms of DLT financial instruments issued on Corda, hence, all reportable information regarding the DLT financial instrument could be implemented in the smart contract representing the DLT financial instrument. Similarly, with respect to information regarding the parties engaged in a Corda transaction in DLT financial instrument, the relevant and desired data could also be stored on the DLT itself, for instance, by collecting it during the parties' onboarding process. In this case, necessary methods to ensure data privacy and confidentiality need to be applied.

117. Another approach would be to store partial data outside of the DLT. This approach is outlined in Figure 10 below. For instance, relevant data, such as reference data for a certain DLT financial instrument or personal information regarding a trading party, could be stored in a database and be linked to Corda transaction data available on the ledger. Both approaches have certain advantages and disadvantages and depend on the respective network and application design. Moreover, further relevant EU regulations with regards to the storage and transfer of personal data may have to be considered when designing a storage solution.

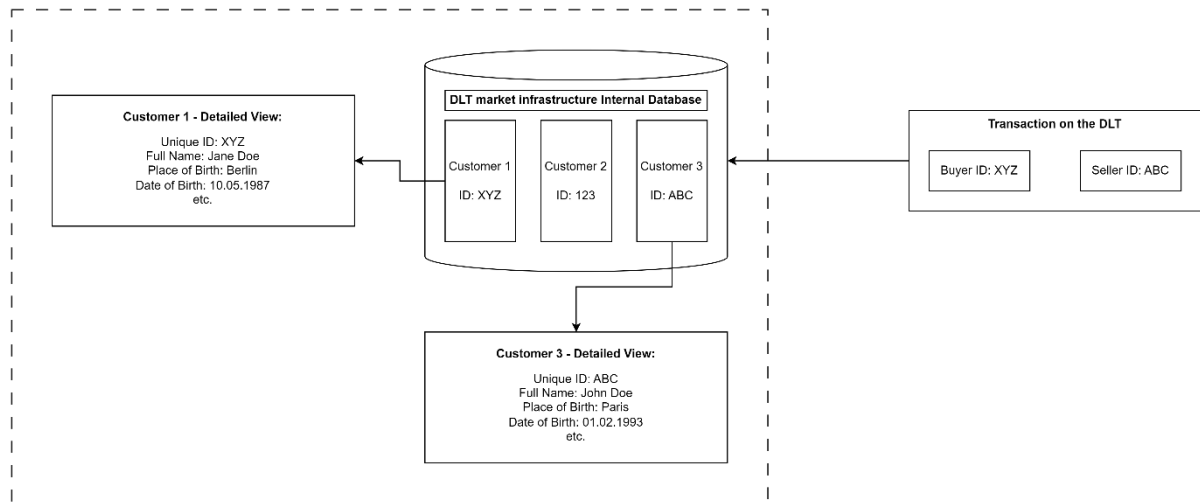


FIGURE 10: OFF-CHAIN DATA STORAGE

118. **Example:** In case DLT market infrastructure operators make use of the Corda DLT for building their respective applications, they may choose one of various approaches for the storage of additional business fields. As outlined, the approaches also have implications regarding network design and should consider levels of data sensitivity. Hence, if all data, including personal data regarding network participants, is stored on the DLT itself, appropriate safeguarding measures ought to be taken. Such measures may include, among other things, encrypting personal data.

g) Correction and cancellation mechanisms

119. In the traditional world, a transaction report for the purpose of RTS 22 reporting can be cancelled by populating Fields 01 (Report status), 02 (Transaction Reference Number), 04 (Executing entity identification code), and 06 (Submitting entity identification code). Such reports are used to cancel RTS 22 transaction reports for non-reportable transactions or transaction reports containing errors. In case, a cancellation was submitted due to an erroneous transaction report being sent in the first place, a correction is submitted.

120. Corda transactions are final meaning the DLT does not natively support the cancellation or correction of Corda transactions. Once Corda transactions have been signed by all required parties, verified as to their validity and uniqueness, and committed to the ledger, they are immutable. This is because if any of the components of the Corda transactions were to change, the associated *txhash* would change as well, which would not be recognised as a valid Corda transaction. This mechanism ensures the technology's authenticity.

121. Although Corda transactions cannot be corrected or cancelled natively, it could be envisioned that, for cases where both parties to a Corda transaction in a DLT financial instrument, and potentially the DLT market infrastructure itself, agree, that the Corda transaction in question contains errors or was not executed the way it was intended, the DLT market infrastructure operator provides certain administrative functions to correct the Corda transaction. Such administrative functions could assist in returning the DLT financial instruments and the e-money tokens or ISO currency used for payment to their original owners and thereby reverse the Corda transaction. Hence, certain business logic could be applied to rectify erroneous Corda transactions.
122. **Example:** In case Party A and Party B engage in a Corda transaction in DLT financial instruments, which turns out to contain errors, the DLT market infrastructure facilitating the Corda transaction could offer certain administrative functions to effectively reverse the Corda transaction. How such an administrative function will be implemented depends on the DLT market infrastructure' discretion, however, various approaches could be envisioned.
123. Typically, the DLT market infrastructure will not be able to access the vaults of the trading parties directly and can therefore not reverse the trade itself. An approach would be for the trading parties to create a corrective and reversed trade via the DLT market infrastructure in order to rectify the erroneous Corda transaction. In this reversed trade, the DLT market infrastructure could ensure that the trade occurs between the two original trading parties. This could be done by implementing a logic in the smart contract requiring that reversed trades must occur between the same counterparties as were involved in the initial trade. In order to ensure the correct reporting of the reversed trade, it should be flagged or made visible as such and clearly refer to the initial Corda transaction.

3.1.4 Gap analysis with respect to RTS 22

124. As mentioned, the RTS 22 specify the content and format, in which in-scope financial transactions ought to be reported to competent authorities. Standardisation of the reporting framework allows for improved supervision and increased transparency within financial markets. RTS 22 fields aim to explore further the nature of a financial transaction and its constituents.
125. While Article 2, paragraphs 2(a) and 3(a) of the RTS 22 respectively define a transaction as the purchase or the sale of a financial instrument, paragraphs 2(b)(c) and 3(b)(c) further include in that definition, the entering into a derivative contract as well as an increase in the notional amount of a derivative contract and the closing out of a derivative

contract as well as a decrease in the notional amount of a derivative respectively.³⁴ The DLT Pilot Regime as outlined in Regulation (EU) 2022/858 of the European Parliament and of the Council of 30 May 2022, however, only applies to shares, bonds, and UCITS, hence making various of the RTS 22 fields superfluous.³⁵

126. More specifically, RTS 22 Fields 32, 40, 42 – 53, 55, 56, and 62 – 64 were pre-assessed by ESMA and declared to generally not pertain to the financial instruments covered by the Regime.
127. The purpose of the gap analysis is to outline which of the Corda DLT's native fields are comparable to fields covered by the RTS 22 reporting regime, which are not comparable, and which native fields are of particular importance to properly supervise trading activities.
128. In the context of the following analysis, it is assumed that the parties to Corda transactions either operate a node or make use of an account provided by a Corda node. In the context of the DLTR, the node or the account can be used to engage in Corda transactions, including Corda transactions in DLT financial instruments. For Corda transactions in DLT financial instruments, the nodes and accounts will trade via a DLT market infrastructure, who will then be tasked with submitting applicable transaction reports. As there is a direct interaction between the trading parties, i.e., the nodes or accounts, and the DLT market infrastructure, as seen in Figure 6, every Corda transaction in DLT financial instruments should result in the production of one transaction report.

3.1.4.1 Fields similar to RTS 22

129. Every Corda transaction will contain information regarding the *txhash*, the applicable *index*, and the *recordedTime*. Out of these three fields, the *txhash* and *recordedTime* can be said to be similar to fields captured by the current RTS 22 reporting schema. While the *txhash* is one of the two components that uniquely identifies a Corda transaction, the *recordedTime* provides information regarding the time of the Corda transaction's commitment to the ledger, i.e., its finalisation.
130. While the *txhash* field could be likened to RTS 22 Field 03 (Trading venue transaction identification code) in the information it conveys, the two have different structures. Therefore, RTS 22 Field 03 cannot simply be analogously applied to a Corda transaction. The RTS 22 namely prescribe the reporting of 52 alphanumeric characters, whereas a *txhash* is composed of 64 hexadecimal characters. Both Corda's *txhash* as well as RTS 22's Field 03, however, serve as unique identifiers. Corda's *txhash* is a unique identifier of

³⁴ https://ec.europa.eu/finance/securities/docs/isd/mifid/rts/160728-rts-22_en.pdf

³⁵ <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32022R0858#d1e966-1-1>

a Corda transaction, whereas RTS 22 Field 03 is a unique identifier of a financial instrument transaction on a specific trading venue.

131. The *recordedTime* field, on the other hand, both contains the same information and follows the same format as RTS 22 Field 28 (Trading date time). Both fields are shown again in Table 1 below.

Corda field	Contained in RTS 22?
Txhash	Information contained can be likened to RTS 22 Field 03 (Trading venue transaction identification code), however differences in format exist.
RecordedTime	Information contained is the same as for RTS 22 Field 28 (Trading date time).

TABLE 1: CORDA FIELDS SIMILAR TO RTS 22

3.1.4.2 Fields not covered in RTS 22

132. As mentioned, every Corda transaction will contain information regarding the *txhash*, the applicable *index*, and the *recordedTime*. Corda's *index* is a field that currently is not captured in the RTS 22 reporting schema. The *index* is the second component, which uniquely identifies a particular Corda transaction. As explained in Paragraph 80, an *index* is used to reference a specific output of a Corda transaction. Hence, its addition to a reporting schema for Corda transactions would likely be helpful as it would provide a more granular view and better understanding of the Corda transaction and the various actions occurring within it as explained in Paragraph 135.

Corda field	Contained in RTS 22?
Index	No.

TABLE 2: CORDA FIELDS NOT CAPTURED BY RTS 22

3.1.4.3 Fields of particular importance

133. As outlined in 3.1.4.1, various fields and further components containing crucial information for regulatory purposes, are included in Corda transactions, however, currently would not be captured by the RTS 22 reporting schema.

134. Firstly, the RTS 22 currently do not carry a field applicable to the *txhash* produced by a Corda transaction. The particular importance of the *txhash* is rooted in its ability to uniquely pinpoint a specific Corda transaction, which has occurred on the Corda network. As the information conveyed by the *txhash* can be likened to the information to be reported under RTS 22 Field 03 (Trading venue transaction identification code), it may be sensible to amend RTS 22 Field 03 to also be able to process a Corda *txhash*. Unlike the current Trading venue transaction identification code (TVTIC) process, where the TVTIC is created and distributed by the trading venue, the *txhash* is computed by the DLT itself. The DLT market infrastructure, therefore, does not have the need to compute or distribute the *txhash* to the involved trading parties.
135. In a similar fashion as with the *txhash*, the *index* provides important information from a regulatory point of view. As mentioned in Paragraph 82, the *index* exactly specifies the output of a Corda transaction. In the case that a DLT financial instrument is exchanged against some form of payment, the index will specify the various parts and thereby the outputs of the Corda transaction. Similar to the *txhash*, the *index* is also computed by the DLT, meaning the DLT market infrastructure does not have the need to compute or distribute the *index* to the involved trading parties.
136. A Corda transaction could, for instance, consist of Party A sending a DLT financial instrument to Party B in exchange for Party B sending a payment to Party A. In such a case, both Party B's receipt of the DLT financial instrument and Party A's receipt of the payment are outputs of the same Corda transaction. Hence, they will be identifiable by the same *txhash*. However, the transaction will contain different indexes (0 and 1 respectively). It thus may make sense to report the *index* to a Corda transaction along with the *txhash* to precisely specify the object defined in the single-sided transaction report. Reporting the *txhash* and the *index* is helpful in that it gives a regulator not only an overall view of the Corda transaction as a whole, but also of the individual activities, e.g., individual trades, it involves, while allowing the regulator to consolidate this information.

Field to be added	Rationale
Txhash	Enables the unique pinpointing of a specific Corda transaction occurring on the network. Either the current TVTIC field could be amended to accommodate for the format of a Corda txhash or an entirely new field specific to the txhash could be added.

Index	Enables the unique pinpointing of the different activities within a single Corda transaction.
-------	---

TABLE 3: NATIVE CORDA FIELDS SUGGESTED TO BE ADDED

137. Table 3 above summarises the native Corda fields suggested to be added to current transaction reports. Furthermore, depending on the structure of the Corda transaction, other fields and information may be important to understand it and its intention. In its Report on the DLT Pilot Regime and as explained previously, ESMA recognises that a DTI is important for various reasons. One of the reasons is that a DTI helps to identify the specific type of e-money token that is being used to settle transactions. Additionally, it enables regulators to create a link and differentiate between traditional financial instruments and their tokenised versions issued on different DLTs.³⁶
138. The latter point is particularly relevant because it is now possible to convert existing financial instruments into digital form so they can be traded on both traditional and DLT-based venues using the same identification number (ISIN). If DLT financial instruments with a DTI are traded on a DLT market infrastructure, it could be useful to include the DTI identifier in transaction reports.
139. A further important reason for the inclusion of the DTI is that it allows for the unique identification of the DLT underlying the DLT financial instrument. Such information provides relevant insights regarding the DLT’s governance structure and helps assess safeguards that may be necessary to prevent market abuse from happening. This is of special importance in case DLT market infrastructure use multiple DLTs to offer their respective services.
140. Similarly, it may be the case that DLT market infrastructure operators may offer settlement of DLT financial instruments traded on their respective platforms to be settled in e-money tokens. In such a scenario, it could be also relevant to include information about the exchange rate between the e-money token and the ISO currency it refers to in transaction reports.
141. Also, it would be important to properly identify any trading parties engaging in Corda transactions in DLT financial instruments via DLT market infrastructures. Depending on the DLT market infrastructure’s network design, nodes may host accounts to be used by further natural persons. In case these accounts are used for the purpose of engaging in Corda transactions in DLT financial instruments, these natural persons must be identified. This can be done via KYC-processes provided by the DLT market infrastructures. It is important

³⁶ https://www.esma.europa.eu/sites/default/files/library/esma70-460-111_report_on_the_dlt_pilot_regime.pdf

to be aware of the trading parties on a Corda network to ensure market integrity and to be able to further investigate any illicit or fraudulent behaviour.

142. It is important to mention that the extra fields compared to the one provided by RTS 22 and discussed in this section do not provide an exhaustive overview over all potentially relevant transaction reporting fields. Which functions and mechanisms are implemented at DLT market infrastructures depends on the DLT market infrastructures' discretion.

143. For instance, some DLT market infrastructures may not offer settlement in e-money tokens or trading in DLT financial instruments identifiable by DTIs. In such cases, the execution of Corda transactions in DLT financial instruments will not contain such newly added fields. On the other hand, DLT market infrastructures may offer further functions not specified in this section that can potentially include extra information that may be relevant from a regulatory point of view. Similarly, created DLT financial instruments could also carry further regulatorily relevant data.

3.1.4.4 Fields relevant for on-chain analysis

144. On-chain analysis can be defined in several ways. Essentially, it is the analysis of activity occurring on a blockchain and, among other things, takes into account blockchain transaction activity.³⁷ On-chain analysis can be a helpful way to understand what is happening on a particular blockchain. It provides investors with market insights and helps them make better investment decisions. By using on-chain analysis, investors can access information about blockchain trends and monitor market sentiment.³⁸

145. Furthermore, on-chain analysis could prove to be a useful tool for regulators. Possible use cases from a regulatory point of view may involve the identification of certain trading patterns that could lead to insights regarding illicit or fraudulent blockchain activities.

146. An example of a trading pattern that could be of interest to a regulator would be if certain network participants purchase large amounts of a particular DLT financial instrument shortly before a major announcement positively affecting the DLT financial instrument's price. Such behaviour could be an indicator of insider trading.

147. A further interesting pattern that could be identified making use of on-chain analytics is that of wash trading to artificially increase trading volumes or pumping and dumping to artificially inflate a DLT financial instrument's price. Moreover, on-chain analysis could

³⁷ <https://primexbt.com/for-traders/what-is-crypto-on-chain-analysis/>

³⁸ <https://www.nansen.ai/guides/what-is-on-chain-analysis-and-why-is-it-useful-for-crypto-traders>

provide for a better understanding of trends or popular investments in the blockchain sphere.

148. Corda's setup as a private and permissioned DLT, however, does not allow for on-chain analysis in the same way that a public blockchain would. A crucial characteristic of effective on-chain analysis is availability of data, for instance, regarding the parties to a transaction occurring on a blockchain or the object of the blockchain transaction itself.
149. As outlined in Paragraph 11, however, there is no global broadcast of Corda transactions. This leads to enhanced data privacy and the inability of third parties to see the details of a Corda transaction. At the same time, however, it also leads to the non-existence of publicly available websites or other popular tools for conducting on-chain analysis.
150. As outlined in Paragraph 16, however, Corda transactions are always stored in the vaults of the parties involved in them. Making use of standard database management systems (DBMS) allows transacting parties to read out all data stored in said vaults and pertaining to the respective Corda transactions. While potentially providing more detailed and granular information regarding specific Corda transactions, it arguably does not provide a broader and more detailed view of the entire network, relevant market trends or trading patterns. Furthermore, if a regulator intended to acquire Corda transaction information beyond the information submitted as part of transaction reports, they could also task DLT market infrastructures with providing it to them, if present.

3.1.5 Conclusion

151. Corda provides for very limited native standardised fields and allows for significant flexibility in how a Corda transaction can be configured and what information it can contain. Therefore, in case a regulator intends to get a comprehensive overview of network activities in the context of the DLTR, clear guidelines will likely be necessary to ensure appropriate market supervision. These guidelines are especially relevant regarding the proper identification of transacted objects, trading parties, as well as price and quantity of DLT financial instruments acquired or disposed of.
152. Furthermore, the integration of a DTI for enhanced identification and understanding of DLT financial instruments as well as e-money tokens used on DLT market infrastructure could also be sensible. Reporting such information to a regulator provides distinct benefits as outlined over the course of this chapter, e.g., identifying the traded object and its underlying DLT. The DTI, however, will not cover the exact reference to a Corda transaction. This is achieved by the *txhash* and *index*. Generally, it will be interesting to see how DLT market infrastructure operators design their respective platforms on the

Corda DLT. Gaining insights and visibility into market practices will allow regulators to work on proper harmonisation and standardisation efforts.

3.2 Ethereum

3.2.1 Background

153. Ethereum was first described in a whitepaper³⁹ by its founder Vitalik Buterin in 2014 before being officially launched in 2015. It is an open source blockchain allowing for the building and operation of various use cases on it, including but not limited to decentralised applications, games, marketplaces, and financial instruments. Further, Ethereum carries its own native (crypto-)currency Ether (ETH). At the time of this report, Ether is only second to Bitcoin in terms of its market capitalisation⁴⁰ and will be further explored later in this section.

154. Smart contracts are of significant importance on the Ethereum network. Tokens, including those mentioned in Paragraph 169 and 170, and thereby DLT financial instruments, are based on specific smart contracts. Smart contracts contain functions, for instance *Transfer*, *Mint* and *Approve* that can emit certain events, which are stored within the transaction receipt, when triggered.⁴¹ Examples of these and other events are given in 3.2.3.4.

155. The blockchain has a single inherent computer, known as the Ethereum Virtual Machine (EVM), embedded in it, a copy of which is kept by all Ethereum network participants, i.e., all nodes. The nodes all agree on the state of the EVM and possess the ability to request it to perform computations. Requests for computations are also called transaction requests.

156. Once such a request is broadcast, the computation is verified, validated, and executed by the other network participants. The computation's execution then changes the EVM's state, which is committed and sent throughout the network. The blockchain further stores the record of all past Ethereum transactions as well as the current state of the EVM.

157. Ethereum transactions can only be initiated by externally owned accounts (EOAs).⁴² EOAs are one of the two types of Ethereum accounts, with the other one being contract accounts.⁴³ EOAs are owned and controlled by individual users, such as natural persons.

³⁹ https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

⁴⁰ <https://coinmarketcap.com/currencies/ethereum/>

⁴¹ <https://goethereumbook.org/events/>

⁴² <https://ethereum.org/en/developers/docs/accounts/#key-differences>

⁴³ <https://ethereum.org/en/developers/docs/accounts/#types-of-account>

They can receive, hold, and send Ether as well as other tokens by way of Ethereum transactions.⁴⁴ They can also interact with the second type of Ethereum accounts, which are contract accounts.

158. Contract accounts are used to deploy and execute smart contracts on Ethereum. Contract accounts are not controlled by any one entity but rather by the logic of the smart contract code. In doing so, they enable a wide variety of use cases, including decentralised applications. An example of a decentralised application enabled by contract accounts and their smart contracts is Uniswap. Uniswap is a decentralised application on which tokens can be traded without further intermediaries.⁴⁵
159. Both kinds of Ethereum accounts have addresses, i.e., unique identifiers that represent the accounts on the Ethereum network. EOAs, however, unlike contract accounts, also have a private key. Private keys on Ethereum provide their respective owners with the ability to have control over access to funds as they prove ownership and control of an EOA on the network. They are also used to sign Ethereum transactions.
160. Ethereum transactions can be sent either to EOAs or contract accounts. In cases where the recipient to an Ethereum transaction is a contract account, the called upon smart contract takes the Ethereum transaction and its data as input to execute certain functions within that contract. For example, in the case of a decentralised application offering trading services, trading processes are handled entirely via smart contracts. If Party A wants to buy a certain token which is traded on such a decentralised application, Party A can initiate an Ethereum transaction, specify the contract account, i.e., the smart contract as its recipient, the token he wants to buy and the amount at which to buy the token, and call a hypothetical *Buy* function implemented in the smart contract. The hypothetical *Buy* function will then, for example, calculate how many of the desired tokens Party A can buy given the amount he specified in the Ethereum transaction, deduct funds from Party A's account, and transfer the acquired tokens to his account.
161. As touched upon in Paragraph 155, Ethereum's network participants are better known as network nodes. Ethereum is designed in a way that the average personal computer can run a node.⁴⁶ However, it is recommended to use dedicated hardware to run nodes in order to minimise node downtime. Nodes can hence be run by anyone and ensure the network's decentralisation and security.
162. Nodes are instances of Ethereum client software connected to other computers running Ethereum software. Together, the multitude of nodes makes up the Ethereum network as

⁴⁴ <https://ethereum.org/en/developers/docs/accounts/>

⁴⁵ <https://uniswap.org/>

⁴⁶ <https://ethereum.org/en/run-a-node/>

such. In total, there are three different types of nodes, each of which consume data differently. While full nodes⁴⁷ store the entirety of blockchain data, assist in block validation and verification, light nodes⁴⁸ only download block headers carrying summarised information about block contents. Lastly, archive nodes build archives of historical states and thus are an effective way of querying historical blockchain data, such as historical account balances.⁴⁹

163. Two different types of clients, namely execution clients and consensus clients, exist on Ethereum.⁵⁰ The execution client executes new Ethereum transactions broadcasted. It further holds the network's most up-to-date version, i.e., the most up-to-date state of all Ethereum data. The consensus client, on the other hand, implements Ethereum's proof-of-stake consensus mechanism. Hence, it is crucial in that it facilitates network agreement regarding the data validated by the execution client. As part of "The Merge", these previously separated layers were connected and integrated into one network.

164. On September 15, 2022, Ethereum underwent "The Merge", shifting the network's way of achieving consensus from proof-of-work (PoW) to proof-of-stake (PoS).⁵¹ Network validators must now stake Ether into Ethereum smart contracts and are responsible for assessing that new blocks are valid and, at times, also creating and propagating new blocks through the network themselves. Validators' staked Ether acts as collateral which can be slashed from the staked balance in case the validator behaves in an undesired manner.⁵² Further, "The Merge" was successful in decreasing Ethereum's energy consumption by 99.9%.⁵³

165. Blocks on Ethereum store Ethereum transactions and reference previous blocks by making use of a hash. Ethereum makes use of Keccak-256 hashes, which, when compared to SHA-256 hashes, are stronger.⁵⁴ Should information within a block change, its accompanying hash would do so as well, as hashes are derived from block data. This is a useful mechanism for fraud prevention as changes to historical blocks would cause all subsequent blocks to become invalid.

166. Blocks and the Ethereum transactions contained within them are strictly ordered and must be agreed on by all Ethereum nodes. For a synchronised state of the network to be possible, blocks are propagated through the rest of the network, once assembled by a

⁴⁷ <https://ethereum.org/en/developers/docs/nodes-and-clients/#full-node>

⁴⁸ <https://ethereum.org/en/developers/docs/nodes-and-clients/#light-node>

⁴⁹ <https://ethereum.org/en/developers/docs/nodes-and-clients/#archive-node>

⁵⁰ <https://ethereum.org/en/developers/docs/nodes-and-clients/#what-are-nodes-and-clients>

⁵¹ <https://www.coindesk.com/tech/2022/09/15/the-ethereum-merge-is-done-did-it-work/>

⁵² <https://blockdaemon.com/products/white-label-validator/ethereum-introduction/#staking>

⁵³ <https://cointelegraph.com/news/the-merge-brings-down-ethereum-s-network-power-consumption-by-over-99-9>

⁵⁴ <https://www.geeksforgeeks.org/difference-between-sha-256-and-keccak-256/>

validator.⁵⁵ The other nodes then add the newly created block to the end of their blockchain before a new validator is randomly selected to assemble the next block. Other randomly selected validators will then vote upon the block's technical validity.

167. Ethereum transactions, as will be explored in more detail in 3.2.3, are cryptographically signed instructions from accounts.⁵⁶ All Ethereum transactions, hence, update the overall state of the Ethereum network and are propagated throughout it. This means, all full and archive nodes have a copy of all transactions that have occurred on Ethereum. Light nodes, on the other hand, only store block headers, which contain information such as block numbers, timestamps, and more. For more detailed information regarding Ethereum transactions, they rely on full nodes. While there are various types of Ethereum transactions, their most rudimentary form refers to the sending of funds, e.g., Ether, from one account to another account.

168. As mentioned before, Ether is the network's native cryptocurrency. It has a wide variety of use cases in the broader DLT ecosystem and, on Ethereum itself, is used for such things as paying for "gas", i.e., transaction fees on the Ethereum network, as well as block proposal and validation. Ether is created in a "minting" process and distributed between the proposer and validators of a block.⁵⁷ Ether can also be "burned" and thereby permanently removed from circulation. Every Ethereum transaction leads to the burning of Ether, as a base gas fee, whose amount is determined by transactional demand on the network, gets destroyed along with the Ethereum transaction.⁵⁸ The process of minting Ether, however, does not lead to its burning.

169. Generally, there are a variety of token standards that find application on Ethereum. Some of the most popular token standards currently are ERC-20 tokens and ERC-721 tokens.⁵⁹ ERC-20 tokens are fungible, meaning an ERC-20 token will always be equal to another ERC-20 token. This makes them especially useful for usage as cryptocurrencies or security tokens. ERC-721 tokens, on the other hand, are the underlying interfaces for so-called non-fungible tokens (NFTs). ERC-721 tokens especially find application in the collectibles or ticketing industry.

170. A further popular token standard is the ERC-1155 standard, which allows for the creation of multi-token contracts. It derives its popularity from the ability to represent various types of digital assets on Ethereum, among those, such assets that are typically

⁵⁵ Ethereum transactions are contained within blocks. Assembling the blocks refers to the process of selecting and including valid Ethereum transactions within a block. Once a block has been assembled, it will be appended to the previous block, thereby creating a chain.

⁵⁶ <https://ethereum.org/en/developers/docs/transactions/>

⁵⁷ Ether is minted, i.e., created as a reward for proposing blocks on the network. There is no alternative mechanism that can be applied to mint Ether.

⁵⁸ <https://ethereum.org/en/developers/docs/intro-to-ether/>

⁵⁹ <https://ethereum.org/en/developers/docs/standards/tokens/>

represented by ERC-20 and ERC-721 tokens. The ERC-3643 standard is a proposed token standard for regulated exchanges.⁶⁰ At the time of writing, however, the Ethereum Improvement Proposal (EIP) is stagnant. Hence, the proposal has seen no activity for at least the preceding six months, making its adoption uncertain.⁶¹

171. Ethereum as a network is constantly being developed through so-called “Ethereum Improvement Proposal” (EIP) processes. Various parties contribute to such development, including but not limited to EIP authors and developers.⁶² Although, there is no central party which is in charge of the network, changes to its core protocol are implemented regularly in order to maintain Ethereum’s stability and security.

172. When it comes to DLTs, there is a difference between on-chain and off-chain governance. While on-chain governance typically involves stakeholder voting, for instance, by making use of a certain governance token, off-chain governance occurs through a process of social discussion. Ethereum itself makes use of off-chain governance, while it should be noted that some of the applications built on top of it utilise on-chain governance.⁶³

173. Proposed changes to the network must undergo the EIP process. Several steps are involved in the decision of whether the proposal is approved or rejected. Should the EIP be approved, it will be scheduled as part of a network upgrade. As everyone on the network must upgrade simultaneously, multiple EIPs tend to be bundled together.⁶⁴

3.2.2 Applied methodology

174. Similar to the approach applied to analyse Corda transactions, the methodology used to analyse Ethereum features multiple pillars.

175. Firstly, a field study explored the essential components of any Ethereum transaction. Secondly, potential applicants from various jurisdictions were identified and contacted to learn about envisioned real-life use cases regarding transaction execution and registration as well as data storage approaches. Lastly, public block explorers such as Etherscan⁶⁵ were used to analyse live applications of financial instruments, which have been launched on Ethereum and EVM-based blockchains.

⁶⁰ <https://eips.ethereum.org/EIPS/eip-3643>

⁶¹ <https://eips.ethereum.org/>

⁶² <https://ethereum.org/en/governance/>

⁶³ <https://ethereum.org/en/governance/#on-chain-vs-off-chain>

⁶⁴ An overview of notable milestones of the Ethereum blockchain can be found here: <https://ethereum.org/en/history/>

⁶⁵ <https://etherscan.io/>

176. As a first step, a regular Ethereum transaction was sent on the Goerli testnet to analyse the fundamental information contained within it. The Goerli testnet is a popular testnet used by developers to refine applications prior to their release on the Ethereum mainnet.⁶⁶

177. This information was supplemented with further learnings derived from potential applicants. Selected potential applicants to the DLT Pilot Regime were contacted to gain a better understanding of how Ethereum and EVM-based blockchains⁶⁷ can facilitate transactions in DLT financial instruments and how the applicable transaction data is registered and stored.

178. To get a holistic picture of live Ethereum use cases, the European Investment Bank's digital bond (ISIN FR0014003521, DTI WGHBLG826) was analysed as to the information contained in its contract.⁶⁸ This was done to explore the possible design and relevant information of a DLT financial instrument's underlying contract on Ethereum.

3.2.3 Main elements of an Ethereum transaction

3.2.3.1 Definition of an Ethereum transaction

179. On Ethereum, transactions are actions initiated by an externally owned account. They are cryptographically signed by the sender and update the state of the Ethereum network.

3.2.3.2 Types of Ethereum transactions

180. Ethereum transactions can be grouped into three categories: (i) *regular* Ethereum transactions; (ii) Ethereum transactions that deploy smart contracts on the blockchain, and; (iii) Ethereum transactions that execute the already deployed smart contracts.⁶⁹ While the three differ in structure and in what they entail, they all update the state of the Ethereum blockchain.

181. A common *regular* Ethereum transaction would be the sending of funds, for instance Ether, from one account to another account. So-called wallets are typically used to connect to accounts and provide an easy-to-use interface to manage one's account balance. Figure 11 below outlines a high-level process flow of a *regular* Ethereum transaction. The world

⁶⁶ <https://cryptometaversealert.com/get-started-with-goerli-testnet-a-beginners-guide/>

⁶⁷ EVM-based blockchains are blockchains that are compatible with the Ethereum Virtual Machine and thus allow for the creation and execution of smart contracts on it. Notable examples of EVM-based blockchains besides Ethereum itself are Polygon, Avalanche, the BNB chain, and many more.

⁶⁸ <https://etherscan.io/token/0x1F3D45E2c6c638A8d6BD1c81c99E6dB6D585EEb#readContract>

⁶⁹ <https://ethereum.org/en/developers/docs/transactions/#types-of-transactions>

state being referred to is defined as the global state of the Ethereum network, which is constantly being updated by Ethereum transactions.⁷⁰

182. A *regular* Ethereum transaction is similar to a wire transfer from one person’s bank account to another person’s bank account, but with the absence of a bank or other third-party intermediary in between the transacting parties. To do so, the *to* field, further explored in Paragraph 224, contains the address of the envisioned recipient. The sample Ethereum transaction conducted on the Goerli testnet as part of this study is an example of such a *regular* Ethereum transaction. Further information regarding it is contained in 3.2.3.5.

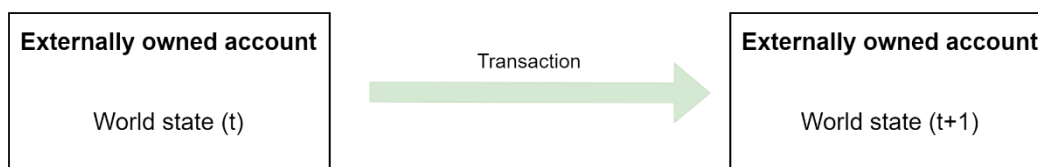


FIGURE 11: EXAMPLE OF A REGULAR ETHEREUM TRANSACTION⁷¹

183. An Ethereum transaction, which aims to deploy a smart contract on the blockchain, is configured slightly differently compared to the *regular* Ethereum transaction. A *deployment* transaction does not specify a particular recipient but is rather sent to the network itself as can be seen in Figure 12 below.⁷² This is done by sending it to the “zero address”, which is specified as *0x0*. In case of such a *deployment* transaction, the *data* field which is further explored in Paragraph 229, will contain the smart contract code to be deployed in bytecode format.⁷³

184. Smart contracts are deployed on Ethereum to make them available to users of the network. The deployment allows other users of the Ethereum network to interact with the smart contract. As explained in Paragraph 185, an Ethereum network participant can interact with a smart contract by sending an Ethereum transaction to it.

⁷⁰ <https://consensys.net/blog/blockchain-explained/ethereum-explained-merkle-trees-world-state-transactions-and-more/>

⁷¹ <https://ethereum.org/en/developers/docs/transactions/>

⁷² <https://ethereum.org/en/developers/docs/smart-contracts/deploying/>

⁷³ Bytecode format refers to the low-level code being generated when a smart contract is compiled from its high-level programming language. In Ethereum, this high-level programming language typically is Solidity. The contract code for the smart contract which is to be deployed is expressed in bytecode format in the *data* field of the deployment transaction.

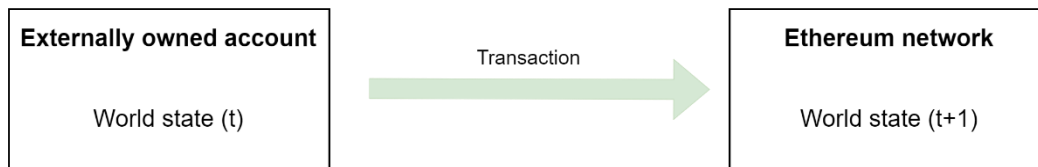


FIGURE 12: EXAMPLE OF AN ETHEREUM DEPLOYMENT TRANSACTION⁷⁴

185. Lastly, Ethereum transactions can be used to trigger functions that are part of smart contracts already deployed on the blockchain. This is outlined in Figure 13 below. An Ethereum transaction directed at a deployed smart contract will specify the smart contract address in the *to* field. To call a certain function contained within it, the necessary Ethereum transaction parameters must be specified.

186. When an Ethereum transaction is sent to a smart contract to trigger a function, encoded events are emitted. Upon emission, these events are included in so-called event logs in the transaction receipt. Events are useful mechanisms to help make sense of an Ethereum transaction. For instance, an Ethereum smart contract could contain a *Trade* function, which, when called, emits a *newTrade* event. This event could be specified to contain the name of the traded DLT financial instrument, its price, and the parties to the Ethereum transaction. Once the Ethereum transaction is finalised, the associated transaction receipt along with the event logs can be analysed to better understand the Ethereum transaction. The transaction receipt is generated upon the Ethereum transaction's successful mining, i.e., its inclusion in a block and contains the fields outlined in 3.2.3.5.

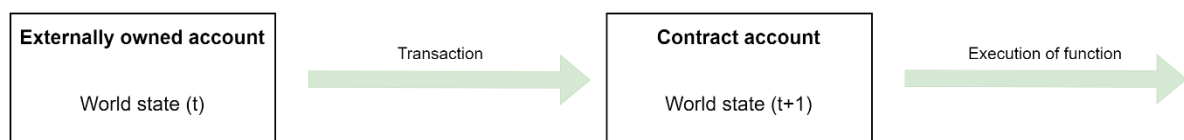


FIGURE 13: EXAMPLE OF AN ETHEREUM TRANSACTION EXECUTING A SMART CONTRACT FUNCTION⁷⁵

3.2.3.3 Established trading processes compared to DLT trading processes

187. As described in 3.1.3.3, current trading and settlement processes involve a multitude of intermediaries. This not only increases complexity but arguably also reduces efficiency. DLT amends this established process by reducing the number of intermediaries and

⁷⁴ <https://kctheservant.medium.com/transactions-in-ethereum-e85a73068f74>

⁷⁵ <https://kctheservant.medium.com/transactions-in-ethereum-e85a73068f74>

stakeholders involved and making it more condensed end-to-end. Figure 6 included in 3.1.3.3 also holds true for Ethereum and outlines amended trading and settlement processes enabled by the technology.

188. In the case that a DLT market infrastructure chooses Ethereum as the underlying technology, how exactly the process will look depends on the design choice of the DLT market infrastructure. Furthermore, what events will be emitted will depend on smart contract specification. An exemplary event that could be emitted upon the successful trade of the DLT financial instrument is *Transfer*.

3.2.3.4 Common Ethereum smart contract standards

189. As stated above in the report, DLT financial instruments are represented through smart contracts on Ethereum and can follow a variety of standards, such as ERC-20 or ERC-721. Depending on the type of smart contract standard used, the produced information in terms of emitted events may differ. Smart contract standards are not driven by a specific use case. Thus, there is no standard that, by its native design, produces the necessary information as necessary for transaction reporting purposes under Article 26 of MiFIR.

190. Generally, there will not be a need for a regulator to gain insight into events emitted by smart contracts. Events facilitate communication and help make sense of Ethereum transactions but generally do not provide for additionally relevant information from a regulatory standpoint that is not already proposed to be included in a transaction report. As will be outlined, most of the common smart contract standards, including the ERC-20, have implemented a *Transfer* event, which will be emitted in case a token transfer was executed successfully. In the case of ERC-20, such event specifies both participants to the transfer along with the amount transferred. While such information (e.g., regarding trading parties and trading amount) is indeed important to a regulator, it is recommended to be included in transaction reports submitted to regulators anyways (see 3.2.4.3). Therefore, it is not required for a regulator to additionally monitor events happening on the Ethereum DLT.

191. A standard ERC-20 token emits the events *Approval* and *Transfer*.⁷⁶ ERC-20's *Approval* event allows another Ethereum address to withdraw funds from the Ethereum address sending the Ethereum transaction. Hence, it is used prior to trading tokens on the network.⁷⁷ Its three fields *from*, *to*, and *value* specify respectively the owner approving the transfer of

⁷⁶ <https://docs.openzeppelin.com/contracts/2.x/api/token/erc20>

⁷⁷ <https://help.1inch.io/en/articles/6147312-token-approvals#:~:text=What%20is%20an%20ERC%2D20,smart%20contract%20like%20adding%20liquidity>.

their tokens, the spender, who has been approved to transfer the tokens, and the value, which is the amount of tokens the owner is approving the spender to transfer.

192. As detailed in Paragraph 190, the *Transfer* event is emitted in cases where tokens are transferred between accounts and includes the sender's as well as the recipient's address along with the amount transferred.⁷⁸

193. Table 4 below outlines concrete scenarios in which the two events may be emitted.

Smart Contract Event	Example
Approval	Party A approves Party B to withdraw a specific amount of tokens from Party A's account.
Transfer	An ERC-20 token is transferred from Party A to Party B.

TABLE 4: EXEMPLARY ERC-20 EVENTS

194. ERC-721 tokens also contain various required events, which are *Transfer*, *Approval*, and *ApprovalForAll*.⁷⁹

195. ERC-721's *Transfer* event specifies the current and the new owner of the token along with the unique ID of the token being transferred.

196. The *Approval* event occurs when the owner of an ERC-721 token approves another party to transfer token ownership to a third party. Such an event will specify the address of the owner, i.e., the address of the network participant approving the ownership transfer, the address of the newly approved account, and the unique ID of the token for which the owner granted approval.

197. Lastly, the *ApprovalForAll* event occurs when the owner of ERC-721 tokens grants or revokes an operator, e.g., a trusted third party, the permission to manage all their ERC-721 tokens. The event will thus specify the address of the owner of the tokens, the address of the operator, who has been granted or revoked approval, and lastly a *true* or *false* value, depending on whether the operator is being granted or revoked permission.

198. Table 5 below outlines concrete scenarios in which the various events may be emitted.

⁷⁸ <https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-Transfer-address-address-uint256->

⁷⁹ <https://docs.openzeppelin.com/contracts/2.x/api/token/erc721#IERC721-Transfer-address-address-uint256->

Smart Contract Event	Example
Transfer	An ERC-721 token is transferred from Party A to Party B.
Approval	Party A approves Party B to withdraw a specific amount of tokens from Party A's account.
ApprovalForAll	Party A approves Party B to withdraw all tokens from Party A's account.

TABLE 5: EXEMPLARY ERC-721 EVENTS

199. As mentioned in Paragraph 170, the ERC-1155 is able to bundle together functionalities typically exhibited by, e.g., ERC-20 and ERC-721 tokens.⁸⁰ This leads to the creation of any combination of fungible or non-fungible tokens.⁸¹ It contains four events that can be emitted, which are *TransferSingle*, *TransferBatch*, *ApprovalForAll*, and *URI*. At the moment the standard especially finds application in blockchain gaming although it may be used in other contexts as well.⁸² However, currently it seems as though this token standard will likely not find widespread adoption as part of the DLTR and hence is not explored further at this point.
200. The ERC-3643 token standard was proposed by tokeny⁸³ for the issue and use of tokenised securities.⁸⁴ Like other smart contracts, it contains certain events that can be emitted. On the top of the ERC-20 fields outlined in Table 4, a token compliant with ERC-3643 must entail the following events: *UpdatedTokenInformation*, *IdentityRegistryAdded*, *ComplianceAdded*, *RecoverySuccess*, *AddressFrozen*, *TokensFrozen*, *TokensUnfrozen*, *Paused*, and *Unpaused*.⁸⁵
201. The *UpdatedTokenInformation* event, as the name suggests, is emitted when the token information is updated and provides information on the name, symbol, decimals, version, and address of the token.
202. The *IdentityRegistryAdded* is emitted when the identity registry for the token has been set and contains the address of the token's identity registry. The identity registry itself is linked to a storage contract that contains a whitelist of identities having undergone

⁸⁰ <https://docs.openzeppelin.com/contracts/3.x/erc1155>

⁸¹ <https://ethereum.org/en/developers/docs/standards/tokens/erc-1155/>

⁸² <https://eips.ethereum.org/EIPS/eip-1155>

⁸³ <https://tokeny.com/about-us/>

⁸⁴ <https://tokeny.com/wp-content/uploads/2020/05/Whitepaper-T-REX-Security-Tokens-V3.pdf>

⁸⁵ <https://github.com/TokenySolutions/T-REX/blob/main/contracts/token/IToken.sol>

appropriate KYC and eligibility checks. KYC and eligibility checks are typically conducted by trusted entities, including common KYC providers. The details of the data acquired regarding the whitelisted identities during the KYC and eligibility checks is not publicly visible. Each security token contains its own identity registry, outlining who is authorised to engage with the token.

203. The *ComplianceAdded* event is emitted when the token compliance has been set. Compliance rules could contain that one token holder may only hold a certain number of tokens or that a token may only be held by a certain number of investors. These compliance rules must be respected all throughout the token's lifecycle and are stored in a dedicated smart contract. Which exact compliance rules are implemented depends on potential legal requirements as well as the token issuer's desires.
204. *RecoverySuccess* is the event emitted when tokens are successfully recovered by an investor. Usually, the loss of private keys to a wallet leads to the loss of the assets stored within the wallet. However, the ERC-3643 supports security token recovery and will provide information regarding the investor's old and new wallet along with the investor's ONCHAINID. While the ONCHAINID contract is stored on the blockchain and investors' ONCHAINIDs are linked to their account addresses, the personal data associated with it will be stored off-chain.
205. Accounts can also be frozen or unfrozen, in which case the *AddressFrozen* event is emitted and specifies whether the event pertains to the freezing or unfreezing. Further, information pertaining to the address being (un)frozen is emitted along with the address of the initiator of the freezing.
206. Similarly, tokens can be frozen and unfrozen. The *TokensFrozen* event is emitted in case tokens on a specific wallet are frozen and specifies the affected wallet and the amount of frozen tokens. The *TokensUnfrozen* event, on the other hand, is emitted when tokens on a specific wallet are unfrozen and again specifies the affected wallet and the amount of unfrozen tokens.
207. Lastly, the *Paused* and *Unpaused* events are emitted when a token is paused or unpaused respectively. Pausing the ERC-3643 token pauses all activity, e.g., trading, in relation to the smart contract. Unpausing the ERC-3643 token unpauses all activity in relation to the smart contract. The event also specifies the address calling the function since this can be triggered by different entities.
208. Table 6 below outlines concrete scenarios in which the various events, excluding the already previously described ERC-20 events, may be emitted.

Smart Contract Event	Example
UpdatedTokenInformation	A DLT financial instrument undergoes a corporate action, e.g., a stock split. To reflect this corporate action, the metadata associated with the token is updated.
IdentityRegistryAdded	An identity registry has been added for the token.
ComplianceAdded	Token compliance has been added, e.g., outlining how many tokens can be held by a single investor.
RecoverySuccess	An investor's lost tokens are successfully recovered and returned.
AddressFrozen	An account is frozen and can no longer engage in Ethereum transactions as it has been involved in fraudulent behaviour.
TokensFrozen	A token is frozen, e.g., as there is evidence that the token has been involved in suspicious Ethereum transactions.
TokensUnfrozen	The reason for the token's initial freezing has been resolved.
Paused	A token is paused, e.g., because there has been a security breach or there is a need for system maintenance.
Unpaused	The reason for the token's initial pausing has been resolved.

TABLE 6: EXEMPLARY ERC-3643 EVENTS

209. In discussions with NCAs, two further token standards were mentioned. These are the ERC-2980 and the ERC-1450. While the ERC-2980 is an interface for asset tokens that is compliant with Swiss law, the ERC-1450, according to its developers, facilitates the

recording of ownership and transfer of securities sold in compliance with the SEC's Securities Act Regulations CF, D, and A.⁸⁶

210. Per its EIP, every ERC-2980 compliant token must implement three events on top of the standard ERC-20 events outlined in Table 4. These are *FundsReassigned*, *FundsRevoked*, and *FundsFrozen*.⁸⁷ With respect to the ERC-2980, a key role is played by the issuer. This issuer has far-reaching permissions, including being able to revoke and reassign funds, i.e., tokens. Issuers are nominated by the owner of the contract and can be the contract owner itself.
211. The *FundsReassigned* event is emitted when funds are reassigned from one account to another account and specifies the amount of the reassigned tokens along with the two involved addresses.
212. The *FundsRevoked* event is emitted when funds are revoked from an account and specifies the affected account along with the amount of revoked funds.
213. Lastly, the *FundsFrozen* account is emitted when an address is frozen and specifies the frozen address.
214. Table 7 below outlines concrete scenarios in which the various events, excluding the already previously described ERC-20 events, may be emitted.

Smart Contract Event	Example
FundsReassigned	An Ethereum transaction was executed erroneously and now the funds are reassigned to the correct holders.
FundsRevoked	A token is revoked from an investor's account, e.g., because said investor is suspected of money laundering or terrorist financing activity.
FundsFrozen	A market participant is accused of engaging in illicit activities and consequently their address is frozen.

⁸⁶ <https://eips.ethereum.org/EIPS/eip-1450#erc-1450--a-compatible-security-token-for-issuing-and-trading-sec-compliant-securities>

⁸⁷ <https://eips.ethereum.org/EIPS/eip-2980>

TABLE 7: EXEMPLARY ERC-2980 EVENTS

215. ERC-1450 tokens must emit three additional events when compared with a standard ERC-20 token. These are *OwnershipTransferred*, *TransferAgentUpdated*, and *PhysicalAddressOfOperationUpdated*. The ERC-20 functions *Approval* and *Transfer* must be implemented to always fail.⁸⁸
216. The *OwnershipTransferred* event is emitted when ownership over the smart contract, i.e., the DLT financial instrument, is transferred from the issuer, i.e., the owner, to a new owner. Hence, unlike the ERC-20 *Transfer* event, this event does not refer to the transfer of a token, e.g., a DLT financial instrument. Instead, it refers to a transfer of smart contract ownership.
217. The *TransferAgentUpdated* event is emitted when the Registered Transfer Agent (RTA) is updated. The RTA, as defined by the ERC-1450, is the only party that may create new securities, transfer, or destroy them. The RTA is a company registered with the SEC and hired by the issuer of the security to serve the role of a record keeper.⁸⁹
218. The *PhysicalAddressOfOperationUpdated* event is emitted when the issuer’s address is updated.
219. Table 8 below outlines concrete scenarios in which the various events may be emitted.

Smart contract event	Example
OwnershipTransferred	Ownership over the smart contract is transferred from Party A to Party B.
TransferAgentUpdated	Previously Party A was the RTA for the tokens but now Party B has been designated as the new RTA.
PhysicalAddressOfOperationUpdated	The issuer of the DLT financial instrument relocates from City A to City B and accordingly, the issuer’s physical address is updated.

TABLE 8: EXEMPLARY ERC-1450 EVENTS

⁸⁸ <https://eips.ethereum.org/EIPS/eip-1450>

⁸⁹ <https://www.startengine.com/blog/introducing-a-new-standard-for-digital-stock-certificates-erc-1450/>

220. It is important to mention that, in some cases, the functions leading to the events emitted by the various ERC-standards described above cannot simply be triggered by any market participant. In such cases, the triggering of certain smart contract functions is restricted to only a few market participants, e.g., the issuers of the tokens.

221. To better understand the kinds of Ethereum transactions occurring on the Ethereum blockchain, events can be “listened” to. Listening to an event means to monitor events, for example by making use of ethers.js, and be notified when they are emitted by smart contracts. The ERC-20’s *Transfer* event, for instance, is emitted anytime someone trades an ERC-20 token. Anyone listening to this event for a specific token will then be notified of the event having been emitted due to a token transfer.⁹⁰ This feature enhances the transparency and auditability of Ethereum transactions, i.e., changes to Ethereum’s state.

3.2.3.5 Structure of an Ethereum transaction

222. Ethereum transactions need to be broadcasted to the entire network and contain, at least, the following pieces of information: A *from* value, which is automatically populated, a *to* value, specifying the Ethereum transaction’s recipient, a *nonce*, a *value*, optionally further *data*, its *gasPrice*, as well as its *gasLimit*. Further, an Ethereum transaction needs to be signed by its initiator.⁹¹ The signing of the Ethereum transaction by the initiator, which is done by making use of the initiator’s private key, can be understood as the initiator authorising it.

223. The account address of the initiator of the Ethereum transaction is contained in the *from* field. In the example of the Ethereum transaction on the Goerli testnet, as shown in Figure 15, the initiator’s account address is 0x1b7aa44088a0ea95bdc65fef6e5071e946bf7d8f. While not being an explicit field in an Ethereum transaction⁹², the *from* address is derived from its public key and starts with “0x”, followed by 40 hexadecimal characters. Block explorers, such as Etherscan, can be used to search all Ethereum transactions involving a given account, as long as the account’s address is known.

224. The recipient of the Ethereum transaction is defined in the *to* field and specifies the address in the to be executed Ethereum transaction. As with the initiator of an Ethereum transaction, the recipient’s address too is derived from its public key and starts with “0x”, followed by 40 hexadecimal characters. Generally, the recipient of an Ethereum transaction can be either an EOA or a contract account. Ethereum itself does not further validate the address.

⁹⁰ <https://moralis.io/how-to-listen-to-smart-contract-events-using-ethers-js/>

⁹¹ https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf

⁹² <https://ethereum.stackexchange.com/questions/133312/geth-client-transaction-object-has-no-from-field>

225. The *nonce* is a further field included in an Ethereum transaction. This field is always a positive whole number, which assists in uniquely identifying an Ethereum transaction. It can also be understood as an Ethereum transaction counter as its value increases by one with each Ethereum transaction sent by a particular address, thereby ensuring every Ethereum transaction is processed only once.⁹³ A nonce of “15” associated with a particular Ethereum transaction, for instance, means that the sender of the Ethereum transaction has thus far sent 16 Ethereum transactions, as the nonce, i.e., the transaction counter starts from “0”.

226. The amount of Ether transferred from the sender of an Ethereum transaction to the recipient is specified in the *value* field. The value is not specified in Ether (ETH) itself, but rather in Wei. Wei is the smallest denomination of Ether and can also be expressed as 1 Wei = 10⁻¹⁸ ETH or 1 ETH = 1,000,000,000,000,000 Wei.⁹⁴ Since not all Ethereum transactions entail a transfer of value, this field is not always populated. Table 9 below provides an overview of the various denominations of Ether.⁹⁵

Unit Name	Value in WEI	Value in ETH
Wei	1	0.000000000000000001
Kwei	1,000	0.0000000000000001
Mwei	1,000,000	0.000000000001
Gwei	1,000,000,000	0.000000001
Twei	1,000,000,000,000	0.000001
Pwei	1,000,000,000,000,000	0.001
Ether	1,000,000,000,000,000,000	1

TABLE 9: DENOMINATIONS OF ETHER

227. The *gasPrice* value specifies the price the initiator of an Ethereum transaction is willing to pay in exchange for gas. In Ethereum, gas is a fee required to conduct Ethereum transactions. Gas fees are priced in small fractions called Gwei of the network’s native currency Ether.⁹⁶ Typically, the more gas one is willing to pay, the more quickly the initiated

⁹³ <https://help.myetherwallet.com/en/articles/5461509-what-is-a-nonce>

⁹⁴ <https://eth-converter.com/>

⁹⁵ <https://moralis.io/gwei-to-eth-how-to-calculate-and-convert-gwei-to-ether/>

⁹⁶ [https://www.investopedia.com/terms/g/gas-ethereum.asp#:~:text=Investopedia%20%2F%20Madelyn%20Goodnight-.What%20is%20Gas%20\(Ethereum\)%3F,\(10%2D9%20ETH\).](https://www.investopedia.com/terms/g/gas-ethereum.asp#:~:text=Investopedia%20%2F%20Madelyn%20Goodnight-.What%20is%20Gas%20(Ethereum)%3F,(10%2D9%20ETH).)

Ethereum transaction will be confirmed. However, in times of low network usage and thus low network demand, it may be the case that no gas needs to be sent at all to get an initiated Ethereum transaction confirmed. Various websites exist today, on which gas prices can be tracked and checked.⁹⁷ The implementation of gas is useful in that it aims to prevent any one party from clogging up the network by continuously sending Ethereum transactions.

228. *gasLimit* is another field inherent to any Ethereum transaction. The value specified in this field pertains to the maximum number of units of gas the initiator of the Ethereum transaction is willing to purchase to complete it. For a *regular* Ethereum transaction, for instance a payment sent from one externally owned account to another externally owned account, this limit is typically 21,000 units of Gwei. More complex Ethereum transactions require more gas as more computational resources are needed. For complex Ethereum transactions, therefore, the *gasLimit* may be higher than 21,000 units.⁹⁸ In such a case, the market participant intending to send this more complex Ethereum transaction should increase their *gasLimit* to ensure the success of the Ethereum transaction. It should be noted that the exact number of gas units required cannot be determined accurately but is rather only estimated.
229. Ethereum transactions also contain an optional *data* field. This field may contain, for instance, metadata regarding an Ethereum transaction. While for *regular* Ethereum transactions, such as the transferral of Ether from one address to another address, the data field is commonly populated with “0x”, it is crucial for the deployment of smart contracts on Ethereum. This was explained in Paragraphs 183 and 184.
230. The *data* field also plays a role in Ethereum transactions sent to an already deployed smart contract. When sending an Ethereum transaction to an already deployed smart contract, the *data* field will typically be populated with the input data, i.e., the necessary parameters, specifying which of the smart contract’s functions is to be called. In case no input data is provided, the smart contract will not be able to execute any of its encoded functions. Figure 14 below shows what input data to an Ethereum transaction may look like. The *MethodID* is the so-called function selector, specifying which of the smart contract’s functions is to be called, while lines [0] and [1] respectively specify the *to* address as well as the *value* of the Ethereum transaction, both in undecoded format.

⁹⁷ <https://ethgasstation.info/>, <https://ethereumprice.org/gas/>

⁹⁸ <https://ethereum.org/en/developers/docs/gas/>

🔍 Input Data:

```
Function: transfer(address to, uint256 value)
MethodID: 0xa9059cbb
[0]: 0000000000000000000000004f6742badb049791cd9a37ea913f2bac38d01279
[1]: 000000000000000000000000000000000000000000000000000000003b0559f4
```

View Input As ▾ 🔗 Decode Input Data

FIGURE 14: EXEMPLARY INPUT DATA TO AN ETHEREUM TRANSACTION⁹⁹

231. As explained in 3.2.1, Ethereum transactions are contained in blocks. These blocks are appended onto another and make up a chain of blocks, i.e., the blockchain. Both the blocks as well as Ethereum transactions contained in them produce *timestamps*. Timestamps specify the date and time at which the block as well as the Ethereum transactions were produced. In addition, it is possible to see which block an Ethereum transaction is included in. This information is available for every Ethereum transaction on the network.

232. Ethereum transactions, as will be explored in more detail in a), also have unique transaction hashes associated with them.¹⁰⁰ Similarly, the blocks they are contained in also possess their respective block hashes and block numbers. These components further ensure the Ethereum network’s integrity.

233. The fields described in this section can be queried through JSON-RPC calls via a client software. A more detailed explanation of this process can be found in the “Report on the DLT Pilot Regime – Study on the extraction of transaction data”. As mentioned, the described fields and pieces of information are the standard components contained or produced in an Ethereum transaction.

234. The majority of the identified potential applicants to the DLT Pilot regime that have selected EVM-based blockchains for their respective use cases as DLT market infrastructures mentioned they expect to have to adhere to the RTS 22 transaction reporting requirements currently in place. To realise this, natively included information is not sufficient and must be extended. Hence, the below paragraphs will outline both the theoretical structure of an Ethereum transaction and supplement this with insights from DLT market infrastructures currently in the process of preparing for licence application.

a) Transaction identifiers

235. Ethereum transactions are included in larger blocks containing a variety of other Ethereum transactions. However, any Ethereum transaction is uniquely identifiable by its

⁹⁹ <https://etherscan.io/tx/0xd0dcbe007569fcfa1902dae0ab8b4e078efe42e231786312289b1eee5590f6a1>

¹⁰⁰ <https://help.safe.global/en/articles/5246870-what-is-the-safe-transaction-hash-safetxhash>

respective *transaction hash*. Also, the accompanying blocks in which Ethereum transactions are included are identifiable by hashes.

236. A *transaction hash* is a 64-character hexadecimal string, which is computed based on the Ethereum transaction's various input parameters. Input parameters are all the components constituting an Ethereum transaction. These are the various fields outlined in 3.2.3.5.

237. Should any of the input parameters to the Ethereum transaction change, the associated *transaction hash* will change the hash being computed as well. Hence, it will be visible all throughout the network that a specific transaction has been altered. This mechanism ensures the immutability of Ethereum transactions and thereby also the overall integrity of the network.

238. Further, as described in Paragraph 231, Ethereum transactions are always included in blocks, which make up the Ethereum blockchain. The *timestamps* associated with Ethereum transactions can be valuable information for regulators as explained in the next paragraph. All Ethereum transactions within a block have the same *timestamp*, which is the time at which the block was produced. Hence, all Ethereum transactions within the same block are executed at the same time.

239. These details can provide regulators with useful insights and help them to better understand the activities taking place on the Ethereum network. When Ethereum *timestamps* are compared to MiFIR transaction data, they provide the same information that is required to be reported under Field 28 (Trading date time) of the RTS 22.

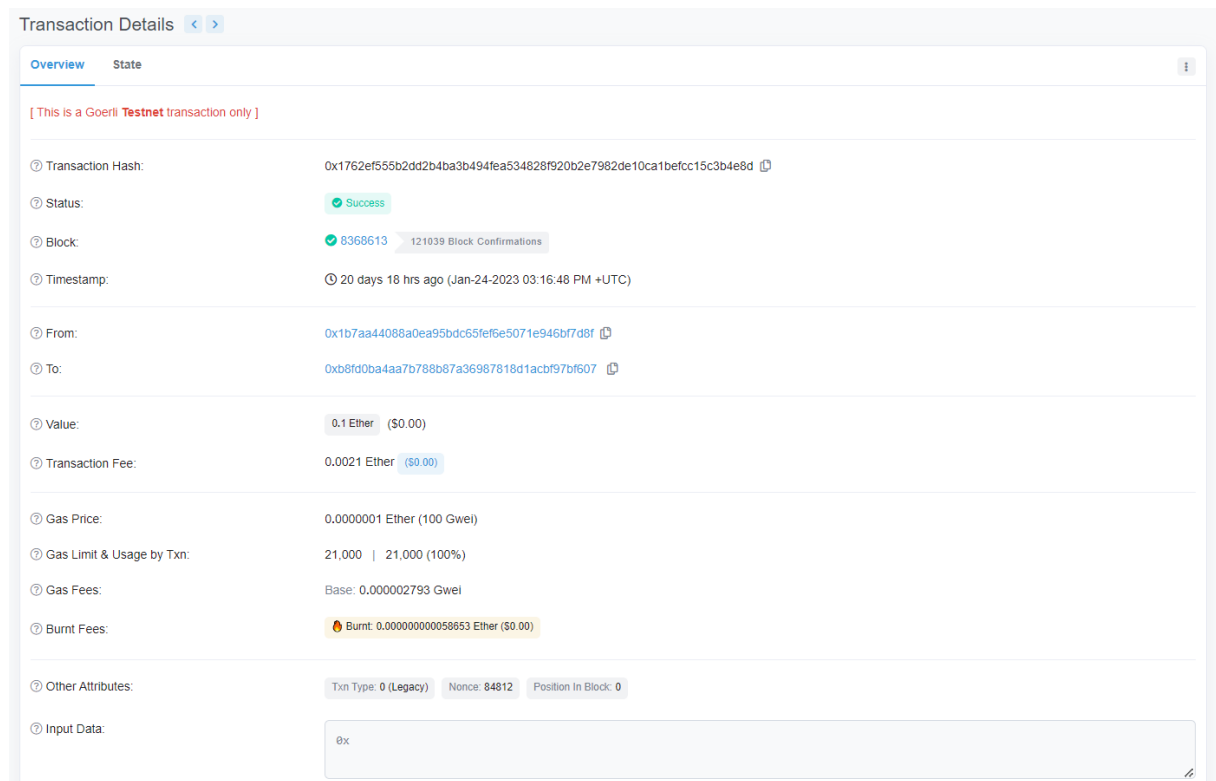
240. Lastly, a *nonce* is also a part of an Ethereum transaction. *Nonces*, as described in Paragraph 225, are positive whole numbers that can only be used once. They are issued by the account initiating the Ethereum transaction and are helpful mechanisms by ensuring the correct order of Ethereum transactions.

241. **Example:** In the case of the sample Ethereum transaction sent as part of this study, the associated *transaction hash* is 0x1762ef555b2dd2b4ba3b494fea534828f920b2e7982de10ca1bfcc15c3b4e8d. As Ethereum transactions are always globally visible and can be analysed, among other approaches, by making use of block explorers, the transaction hash can now be searched for to extract all available information regarding the sample Ethereum transaction.

242. Searching for this Ethereum transaction on Etherscan, a common block explorer, produces the results shown in Figure 15 below. In addition to displaying the *Transaction Hash*, other details are also available such as the *Status* of the Ethereum transaction, the *Block* it is included in, and the date and time it was created (*Timestamp*). This information

can provide a better understanding of the Ethereum transaction. The sample Ethereum transaction sent was successful as can be seen by the *Status* and is included in *Block* 8368613. Ethereum transactions always have the same *timestamps* as their respective blocks.¹⁰¹ In this case, the Ethereum transaction and its block were produced on 24 January 2023 at 03:16:48 PM +UTC.

243. Furthermore, the two parties to the Ethereum transaction are also specified (*From; To*). Party identifiers will be looked at in more detail in b). Additional information included in the Ethereum transaction regard its value and transaction fee expressed in Ether (*Value; Transaction Fee*), the gas price (*Gas Price*), the gas limit allocated to the Ethereum transaction, and the amount actually used (*Gas Limit & Usage by Txn*), the gas and burned fees (*Gas Fees; Burnt Fees*), as well as other attributes (*Other Attributes*) including the nonce and potential input data (*Input Data*). Overall, the global visibility of Ethereum activities makes the supervising of Ethereum transactions very transparent by publicly displaying various of the Ethereum transaction’s basic components.



Transaction Details < >

Overview State

[This is a Goerli Testnet transaction only]

Transaction Hash: 0x1762ef55b2dd2b4ba3b494fea534828f920b2e7982de10ca1befcc15c3b4e8d

Status: Success

Block: 8368613 121039 Block Confirmations

Timestamp: 20 days 18 hrs ago (Jan-24-2023 03:16:48 PM +UTC)

From: 0x1b7aa44088a0ea95bdc65fef6e5071e946bf7d8f

To: 0xb8fd0ba4aa7b788b87a36987818d1acb97bf607

Value: 0.1 Ether (\$0.00)

Transaction Fee: 0.0021 Ether (\$0.00)

Gas Price: 0.0000001 Ether (100 Gwei)

Gas Limit & Usage by Txn: 21,000 | 21,000 (100%)

Gas Fees: Base: 0.000002793 Gwei

Burnt Fees: Burnt: 0.00000000058853 Ether (\$0.00)

Other Attributes: Txn Type: 0 (Legacy) Nonce: 84812 Position In Block: 0

Input Data: 0x

¹⁰¹ <https://medium.com/coinmonks/the-concept-of-time-in-ethereum-f26630a616cd>

FIGURE 15: SAMPLE ETHEREUM TRANSACTION ON THE GOERLI TESTNET¹⁰²

b) Party identifiers

244. As mentioned, any Ethereum transaction will contain identifying information regarding its initiator and its recipient, regardless of the Ethereum transaction's use case. This information is encrypted in account addresses, which are derived from the participants' respective public keys. As the *from* field in an Ethereum transaction always refers to its initiator, it will specify the seller in the scenario that a DLT financial instrument was disposed of, whereas the *to* field would specify the buyer. Talking to potential applicants in the process of the setting up DLT market infrastructures to operate under the DLTR further outlined what possible identification processes of parties to an Ethereum transaction may entail.
245. Regarding the identification of the parties involved in Ethereum transactions in DLT financial instruments, it should be noted that Ethereum makes use of a global broadcast model regarding all Ethereum transactions. Hence, certain details regarding Ethereum transactions are publicly available and can be seen by uninvolved third parties. Consequently, it is crucial that, despite this native mechanism of the DLT, the requirements of Regulation (EU) 2016/679¹⁰³ and other potentially relevant regulations are appropriately considered.
246. **Example:** Using the case of the sample Ethereum transaction conducted on the Goerli testnet (see Figure 15 above), information regarding its parties, i.e., the sender and recipient of the funds, is specified in the *from* and *to* fields. In the above scenario, the party by the address 0x1b7aa44088a0ea95bdc65fef6e5071e946bf7d8f sends 0.1 Ether to the party by the address 0xb8fd0ba4aa7b788b87a36987818d1acbf97bf607.
247. Supplementing these account addresses with further personal information during the onboarding process as is currently envisioned by the potential applicants that have been contacted, will allow for the provision of further, more detailed personal information in transaction reports. To do so, DLT market infrastructures' KYC processes can be used to match personal information to account addresses. KYC information collected could then allow for the provision of additionally relevant information as required by the current RTS 22 transaction reporting schema under MiFID II/MiFIR.
248. To ensure compliance regarding the personally identifiable data obtained during the KYC process, such data could be stored off-chain along with a unique identifier for every market participant that has successfully undergone the KYC process at a DLT market

¹⁰² <https://goerli.etherscan.io/tx/0x1762ef555b2dd2b4ba3b494fea534828f920b2e7982de10ca1befcc15c3b4e8d>

¹⁰³ <https://eur-lex.europa.eu/eli/reg/2016/679/oj>

infrastructure. The unique identifier must then be present in all Ethereum transactions in DLT financial instruments via that DLT market infrastructure for the Ethereum transaction to be executed successfully. Such an approach was confirmed by the contacted potential applicants.

c) Object identifiers

249. Generally, an Ethereum transaction does not natively provide for a field in which the transacted object is specified, for instance by making use of an ISIN, where available, and a DTI. In case such a field is desired or even required from a regulatory standpoint, it can be implemented in the respective smart contracts associated with the DLT financial instrument or be enriched post-trade by the report-submitting entity. As mentioned in Paragraph 94, a DTI provides further technical attributes regarding a traded token. With respect to the bond issued by the European Investment Bank, the DTIF Registry contains further information included in Figure 16.

WGHBLG826	
Token Type: Auxiliary Digital Token	
Normative Attributes	Informative Attributes
Auxiliary Digital Token Mechanism: ERC-20	Long Name: European Invest/Zero Cpn Bd
Auxiliary Digital Token Distributed Ledger: X9J9K872S	Short Name: FR0014003521
Auxiliary Digital Token Technical Reference: 0x1ff3d45e2c6c638a8d6bd1c81c99e6db6d585eeb	Unit Multiplier: 1
Underlying Asset External Identifiers	
ISIN: FR0014003521	
Linked Tokens Informative Attributes	
^ X9J9K872S	
Short Name: ETH	
Long Name: Ethereum	
Public Distributed Ledger Indicator: true	
Reference Implementation URL: https://github.com/ethereum	
Unit Multiplier: 1,000,000,000,000,000,000	

FIGURE 16: TECHNICAL ATTRIBUTES OF THE EIB BOND AS SPECIFIED IN ITS DTI

250. The normative attributes listed outline the required information to uniquely identify a specific token on a DLT. In this case, the EIB bond is based on the ERC-20 token standard, resides on the Ethereum DLT, and has 0x1ff3d45e2c6c638a8d6bd1c81c99e6db6d585eeb as its address. The informative attributes are not required information and can be extended or reduced.

251. Furthermore, it should be mentioned at this point that only a minority of the potential applicants are planning to obtain the DTI for the DLT financial instruments traded on their

platform. Therefore, it may also be sensible to report on the smart contract address itself. Further, some of them mentioned that there are currently no plans to integrate already existing off-chain financial instruments on their platform and thereby make them available to be traded on the blockchain.

252. **Example:** As mentioned, there is no native implementation of an object identifier as part of an Ethereum transaction. Therefore, such an identifier would have to be specified, which can be achieved in several ways. For instance, the identifier can be included in the smart contract upon the DLT financial instrument's issue.

253. To do so, the ISIN or DTI of a DLT financial instrument could be specified during the DLT financial instrument's creation. During the creation, of the DLT financial instrument, i.e., the smart contract representing the DLT financial instrument, a constructor can be used. In Solidity, a commonly used programming language, a constructor can be used to initialise state variables in a smart contract.¹⁰⁴ Taking the example of an ERC-20 token representing a DLT financial instrument, the token can be assigned the name during its construction. This name could be populated with the DLT financial instrument's ISIN or DTI. Other attributes to be specified in this phase are the DLT financial instrument's total supply, its symbol, and more. Figure 17 below shows how the token name and symbol of an ERC-20 token could be defined.

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.9;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract ACME_Equity is ERC20 {
7
8     // Define ERC20-Token name and symbol
9     constructor() ERC20("FR1234567890", "ACME") {}
10
11     // ... Contract Code
12 }

```

FIGURE 17: ERC-20 TOKEN CONSTRUCTOR EXAMPLE

254. Also, other ways of identifying the object can be utilised. One other way would be the integration of an oracle. Oracles¹⁰⁵ can also be used on Ethereum. That way, off-chain data (i.e., external to the DLT) is brought into the blockchain and made available for usage. In theory, digital twins of off-chain financial instruments could be created on the Ethereum blockchain. The oracle could then be used to provide relevant information about the

¹⁰⁴ <https://cryptomarketpool.com/constructor-in-solidity-smart-contracts/#:~:text=A%20constructor%20in%20Solidity%20is,used%20to%20set%20initial%20values.&text=A%20constructor%20can%20be%20either%20public%20or%20internal.>

¹⁰⁵ <https://ethereum.org/en/developers/docs/oracles/>

financial instrument tradable off-chain to its digital representation on-chain. This information can, among other things, entail its ISIN or other identifiers. However, it should be noted that such an implementation was not mentioned in discussions with potential applicants to the DLT Pilot Regime.

d) Price, quantity, and conversion mechanisms

255. On the topic of price, Ethereum does natively possess a *value* field, as briefly discussed in Paragraph 226. In cases, where an Ethereum transaction consists of the sending of units of Ether from one address to another address, the *value* field will be populated with the respective amount transferred. However, this mechanism only applies to Ethereum transactions denominated in Ether and can thus not be applied to Ethereum transactions denominated in any other tokens, e-money tokens, or crypto-assets other than Ether.

256. For instance, in cases where the payment accompanying an Ethereum transaction is conducted in an asset that is not Ether but a licenced e-money token, another mechanism to incorporate the price will have to be implemented. As the DLT Pilot Regime does not cover payments in Ether or in crypto-assets other than e-money tokens, such an alternative approach must be found.

257. If a DLT financial instrument is to be restricted in a way determining that it can only be acquired using central bank money or licenced and regulated e-money tokens, such a specification must be implemented in the smart contract of the DLT financial instrument. Similarly, in cases where the quantity of a DLT financial instrument acquired or disposed of must be specified in the number of units as prescribed by RTS 22 transaction reporting specifications, this mechanism must also be implemented in the respective smart contracts.

258. In case there is a need for a way to convert licensed and regulated e-money tokens to ISO currencies, then DLT market infrastructures must have a mechanism to provide this function. The network itself does not have the capability to do this natively, so it must be implemented separately. Current discussions with potential DLT market infrastructure operators in EVM-based blockchains show that there is some interest in accepting licenced and regulated e-money tokens as means of payment on their respective DLT market infrastructures. Such e-money tokens could be issued either by the DLT market infrastructure itself or a third-party provider connected to the DLT market infrastructure, provided appropriate licenses are in place. Further, e-money tokens could also be assigned a DTI and thus provide for further technical information regarding the token and the underlying DLT.

259. **Example:** In practice, it seems as though both payment options are of interest to potential applicants to the DLT Pilot Regime. This means, e-money tokens to be used for

Ethereum transactions in DLT financial instruments could be issued by the DLT market infrastructure itself or be provided by a third-party the DLT market infrastructure partners with. If payments in e-money tokens are accepted on a DLT market infrastructure, it may be sensible to implement a conversion mechanism for e-money tokens to be converted back to ISO currency. These approaches were confirmed by the contacted potential applicants.

e) Other attributes

260. As discussed in previous paragraphs, some of the natively implemented Ethereum transactions fields may provide relevant information by uniquely identifying an Ethereum transaction or the parties to an Ethereum transaction. Other natively present fields do not serve as transaction, object, price, or quantity identifiers. These are the *gasPrice*, *gasLimit*, and *data* described and explained in Paragraphs 227 – 229. While *gasPrice* and *gasLimit* pertain to the fees associated with an Ethereum transaction, the *data* field is mainly used for the deployment of smart contracts on the network as well as the interaction with them. If an Ethereum transaction is not sent to a smart contract, the *data* field will be populated with *0x*.

261. Oracles can actively be used to integrate further information on the Ethereum ecosystem that may not otherwise be available. Making use of oracles opens up a variety of new opportunities to incorporate additional information within an Ethereum transaction. Furthermore, Ethereum smart contracts can be specified in such a way as that they may provide more detailed information, e.g., regarding the transacted object. This is explained in Paragraph 253.

262. **Example:** Additional information provided through oracles can entail a wide scope of things. Besides providing accurate instrument identifiers as mentioned in Paragraph 254, they could provide additional information regarding the issuer of a DLT financial instrument (see Paragraph 114). Moreover, the smart contracts representing the DLT financial instrument could be written in such a way as that further attributes regarding the instrument itself are encoded within it. An example would be for the smart contract to specify certain restrictions regarding holding or voting rights of the DLT financial instrument.

f) Storage of additional business fields

263. Regarding the storage of additional business fields, various approaches are possible. One of the potential applicants interviewed is planning to only store what they define as the “core transaction” on the DLT itself. Per the potential applicant’s definition, the “core transaction” consists of the identification of the two parties involved in it along with the associated price and quantity.

264. The core transaction will then be linked to further data residing off-chain by making use of an identifier. In this example, off-chain data would therefore be, among other things, reference data regarding the traded DLT financial instrument. Making use of the identifier linking the two data sets allows for a comprehensive understanding of the Ethereum transaction in DLT financial instruments that has occurred. Assuming regulatorily important data is partially stored off-chain, a regulator would not have immediate or direct access to it. Potentially, they could request such data from the DLT market infrastructure or a dedicated interface must be developed. Data extraction and provision methods are explored further in the “Report on the DLT Pilot Regime – Study on the extraction of transaction data”.
265. In practical terms, on-chain data refers to data that is written to the blockchain through a smart contract. Besides functions that can read on-chain data, smart contracts also contain functions that write data to the blockchain. Taking the example of a smart contract representing an ERC-20 token, a function writing to the chain is *Transfer*, which in turn emits the *Transfer* event. As outlined the *Transfer* event is emitted anytime the ERC-20 token is transferred from one account to another account. Hence, it would be emitted in case an ERC-20 token which represents a DLT financial instrument is transferred from a buyer to a seller.
266. A further approach that could be envisioned is to store other subsets of the data on the blockchain itself and link it to off-chain data. Moreover, it could also be a sensible approach to store all data pertaining to an Ethereum transaction on the blockchain itself. Such an approach, however, was not mentioned by the potential applicants engaged in discussions with.
267. **Example:** Generally, there is currently no standardised approach between the potential applicants engaged in discussions with regarding the topic of how additional business fields will be stored. The opportunities to go about this range from storing all applicable data directly on the Ethereum blockchain to only storing minimal information pertaining to the Ethereum transaction on Ethereum itself. Currently, there is no smart contract standard that natively provides all the relevant attributes necessary under MiFIR RTS 22 and therefore could be used to store all relevant DLT financial instrument data on the DLT, meaning the additional of an external database to supplement on-chain information may be sensible.
268. Moreover, such an additional database will also be necessary for the storage of personal information regarding the parties to a DLT financial instrument transaction. Figure 10 outlines the scenario that a DLT market infrastructure stores some personal data in an internal database off-chain and uses a unique identifier for data linkage.

269. In absence of a (binding) guideline, the exact implementation depends on the DLT market infrastructure operators themselves. As already noted in Paragraph 245, an important factor in deciding between the various approaches and storage mechanisms may be other regulations applicable in the European Union and pertaining to the protection of personal data.

g) Correction and cancellation mechanisms

270. Paragraph 119 explains the logic behind cancellations and corrections regarding RTS 22 transaction reports. The Ethereum blockchain is immutable in that anything that occurs on it is final. In practice, potential applicants seem to be willing to implement a function to rectify erroneous Ethereum transactions. Currently, discussions suggest that such a correction function, if implemented, will be handled by the potential applicants themselves through an administrative process. This administrative process takes care of both the delivery and payment leg based on a pre-defined sequence.

271. Said administrative process will occur post-trade and allow an administrator, that is part of the platform and possesses the necessary privileges, to correct an Ethereum transaction in DLT financial instruments by applying business logic. Doing so, will create a new public event visible to everyone and specify that an administrator made a transfer. The publicity and visibility hence are envisioned to make it difficult to abuse this function.

272. **Example:** Under MiFIR, there currently exist a variety of reasons for the cancellation of financial transactions. Usually, financial transactions under RTS 22 are cancelled because they contain errors, such as incorrect quantity or price specifications (e.g., fat fingers). If both counterparties to the financial transaction agree that the trade is flawed, it will typically be reversed. How exactly the reversal will occur depends on the status of the RTS 22 transaction. If the RTS 22 transaction has already been settled, it may be necessary to reverse the settlement and initiate a new and corrected RTS 22 transaction. If no settlement has occurred yet, there may be the option of simply cancelling the RTS 22 transaction and initiating a new and corrected RTS 22 transaction.

273. In the context of a transaction in a DLT financial instrument, such a correction could be initiated if all parties to the Ethereum transaction in DLT financial instruments uniformly agree that it should be reversed and thereby corrected. In such a case, the DLT market infrastructure could be tasked with having to take care of the actual reversal.

274. Potentially, the DLT market infrastructure may be granted access to the accounts or DLT financial instruments of the parties involved in the Ethereum transaction and initiate an Ethereum transaction to send the DLT financial instrument back to its original owner as well as return the funds used to purchase the DLT financial instrument to the buyer. This undertaking could then emit a specific event identifying the Ethereum transaction as an

administrative correction. As mentioned by the potential applicant, this seems to be a feasible and sensible approach. The potential applicants further mentioned such a function likely needs proper guidelines for the function not to be abused or applied carelessly.

275. As the new Ethereum transaction, i.e., the administrative correction, will produce a new *transaction hash*, a way to link it to the initial Ethereum transaction must be found for a regulator receiving the transaction report to undoubtedly know which Ethereum transaction has been cancelled. A way to do so, is to require DLT market infrastructures to produce an internal identifier per Ethereum transaction and submit it as part of a transaction report. This identifier would then have to be the same for the initial Ethereum transaction as well as potential future administrative corrections.

3.2.4 Gap analysis with respect to RTS 22

276. Although several RTS 22 fields are not natively captured in a regular Ethereum transaction, Ethereum smart contracts can be specified in such a way, as to include RTS 22 information, where applicable. As discussed in Paragraph 126, however, not all 65 fields currently covered by the RTS 22 will find application to the DLT financial instruments traded as part of the Pilot Regime and thus not all of them ought to be taken into account when specifying guidelines for the creation of smart contracts.

277. The below sections will outline native Ethereum fields similar to the current RTS 22 transaction reporting schema, deviations between Ethereum transactions and the RTS 22 reporting schema, and further provide suggestions regarding fields that are of particular importance when it comes to supervising trading activity taking place on Ethereum and EVM-based blockchains. Lastly, an overview of particularly relevant fields for on-chain analysis will be provided.

278. In the context of the following analysis, it is again assumed that the parties to an Ethereum transaction have an account on the network. In the context of the DLTR, the account can be used to engage in Ethereum transactions, including Ethereum transactions in DLT financial instruments. For Ethereum transactions in DLT financial instruments, the accounts will trade via a DLT market infrastructure, who will then be tasked with submitting applicable transaction reports. As there is a direct interaction between the trading parties, i.e., the accounts, and the DLT market infrastructure, as seen in Figure 6, every Ethereum transaction in DLT financial instruments should result in the production of one transaction report. The presence and relevance of the fields explored in the following sections remains the same for all kinds of Ethereum transactions in DLT financial instruments.

3.2.4.1 Fields similar to RTS 22

279. Ethereum transactions natively contain certain pieces of information that could be likened to fields captured by the current RTS 22 transaction reporting regime in the information they convey. Some, like the *timestamp*, even contain the same information as RTS 22 fields and are specified in the same format.
280. The *timestamp*, which is described in Paragraph 231, specifies the time at which the block the Ethereum transaction is contained in was produced. The time, at which the block is produced is also the time, at which the included Ethereum transactions themselves are produced. Therefore, the *timestamp* precisely specifies the production time of an Ethereum transaction.
281. The *transaction hash* of an Ethereum transaction is a unique identifier of an Ethereum transaction. Therefore, it can be likened to RTS 22 Field 03 (Trading venue transaction identification code) in the information it conveys. The two fields, however, have different structures. Paragraph 287 provides more information about the difference in format and its impact.
282. A further example would be the *from* and *to* fields discussed at multiple points over the course of this chapter, which could be likened to RTS Fields 16 (Seller identification code) and 07 (Buyer identification code) of the current RTS 22 transaction reporting schema. In the context of an Ethereum transaction in DLT financial instruments, the *from* field will thus specify the seller, whereas the *to* field will specify its buyer.
283. However, the information conveyed by these fields also largely depend on the context of the applicable Ethereum transaction. In the case that two parties engage in a *regular* Ethereum transaction, in which Ether is sent from one account to another account, the *from* field specifies the sender of the Ether rather than its seller. Similarly, the *to* field would then specify the Ether's recipient, not its buyer. Hence, the context of an Ethereum transaction can also be argued to be relevant to understand its overall purpose. Table 10 outlines Ethereum's native fields that can be compared to current RTS 22 fields.

Ethereum field	Contained in RTS 22?
Transaction Hash	Information contained can be likened to RTS 22 Field 03 (Trading venue transaction identification code), however differences in format exist.

Timestamp	Information contained is the same as for RTS 22 Field 28 (Trading date time).
From	Information contained can be likened to RTS 22 Field 16 (Seller identification code), however differences in format exist. Also context-dependent.
To	Information contained can be likened to RTS 22 Field 07 (Buyer identification code), however differences in format exist. Also context-dependent.

TABLE 10: ETHEREUM FIELDS SIMILAR TO RTS 22

3.2.4.2 Fields not covered in RTS 22

284. Ethereum transactions natively contain certain pieces of information currently not covered by the RTS 22 transaction reporting schema. It could be discussed if they are necessary for regulatory purposes. Table 11 outlines the basic Ethereum fields and pieces of information contained in an Ethereum transaction and not represented in the current RTS 22 transaction reporting schema.

Ethereum field	Contained in RTS 22?
Nonce	No.
Value	No.
Data	No.
Gas price	No.
Gas limit	No.

TABLE 11: ETHEREUM FIELDS NOT CAPTURED BY RTS 22

3.2.4.3 Fields of particular importance

285. As outlined over the course of this chapter, various Ethereum fields and other pieces of information natively exist, which provide insights into the nature of an Ethereum transaction. Hence, they may be of particular importance to properly supervise trading activity according to ESMA and NCA mandates.

286. Firstly, a crucial factor to properly supervise trading activity is the *transaction hash*, which is produced as part of an Ethereum transaction. The 64-character hexadecimal string uniquely identifies an Ethereum transaction and can be used as a starting point to analyse Ethereum transactions further. Its mechanism for computation, which is based on the input parameters of its associated Ethereum transaction, further is important in keeping the Ethereum network secure.
287. The *transaction hash*, as discussed in Paragraph 284, can be likened to RTS 22 Field 03 (Trading venue transaction identification code) based on the information it conveys. A definition and explanation of RTS 22 Field 03 (Trading venue transaction identification code) is provided in Paragraph 134. To capture the information contained within an Ethereum transaction hash, the TVTIC’s current format would have to be extended. Currently, the TVTIC field can be populated by up to 52 alphanumerical characters, while the Ethereum *transaction hash* contains 64 hexadecimal characters. Hexadecimal characters include the numbers from 0-9 and letters from A-F. Therefore, an Ethereum *transaction hash* has twelve additional characters when compared with the current TVTIC field, while also making use of a different numbering system. For transaction reporting purposes under the DLT Pilot Regime, either an amendment to the current TVTIC field or the inclusion of a field specifically for the reporting of an Ethereum *transaction hash* is recommended.
288. Furthermore, information regarding the two parties to an Ethereum transaction is also crucial from a regulator’s point of view. In Ethereum, this information is contained in the *from* and *to* fields by specifying the respective wallet addresses of the parties. Knowing the parties to an Ethereum transaction in DLT financial instruments provides valuable information to regulators as it, for instance, allows for better market surveillance. It would therefore be sensible to include this in transaction reports.
289. Regulators could use such information to monitor the various activities of market participants more closely. This may lead to improved detection of potentially fraudulent behaviour, including but not limited to insider trading or other forms of market manipulation. On a broader scheme, hence, appropriate party identification may help manage and reduce risks associated with the trading of DLT financial instruments. Table 12 below outlines the native Ethereum fields proposed to be added to a transaction reporting schema.

Field to be added	Rationale
Transaction Hash	Enables the unique pinpointing of a specific Ethereum transaction occurring on the network. Either the current TVTIC field could be amended to accommodate for the format of a Ethereum transaction hash or

	an entirely new field specific to the transaction hash could be added.
From	Enables the unique pinpointing of the sender/seller involved in an Ethereum transaction. Such information could be complemented with further KYC-data, if acquired by the DLT market infrastructure during client onboarding.
To	Enables the unique pinpointing of the receiver/buyer involved in an Ethereum transaction. Such information could be complemented with further KYC-data, if acquired by the DLT market infrastructure during client onboarding.

TABLE 12: NATIVE ETHEREUM FIELDS SUGGESTED TO BE ADDED

290. As the previously outlined fields and pieces of information merely cover those that are natively implemented in Ethereum transactions, they do not consider how smart contracts and thereby applications on Ethereum may be designed, including the additional reference data they may provide. Should smart contracts be designed in a way that DLT financial instruments traded on the respective platforms built on Ethereum or another EVM-based blockchain are identified by making use of an ISIN or a DTI, it may be appropriate to also capture this information and provide it to the respective NCAs as part of submitted transaction reports.

291. Such instrument identifiers further provide a plethora of insightful information and help make sense of market activities. ISINs, for instance, contain information regarding the country of the issuer of the instrument by including country code prefixes in compliance with ISO 3166-1.¹⁰⁶ DTIs, on the other hand and among other things, provide information on the smart contract standard used to represent the DLT financial instrument.¹⁰⁷ As mentioned in Paragraph 251, it is thus also sensible to report the smart contract address of a DLT financial instrument itself.

292. Information regarding ISINs and DTIs traded, for instance, allow NCAs and other regulatory bodies to monitor volumes as well as movements regarding these DLT financial instruments. Such transparency may prove helpful in detecting suspicious activities and

¹⁰⁶ <https://committee.iso.org/files/live/sites/tc68/files/Robin%20Doyle/What%20is%20ISIN-Final.pdf>

¹⁰⁷ <https://dtif.org/registry-search/>

thereby be effective in reducing market abuse. They can further be a helpful starting point regarding on-chain analysis further explored in 3.2.4.4.

293. Similarly, as touched upon in d), although there is a field conveying the value of an Ethereum transaction, it is only applicable in cases where the value is expressed in Ether. Therefore, arguably no field conveying the price of an Ethereum transaction in DLT financial instruments, similar to RTS 22 Field 33 (Price) exists. Furthermore, there is no natively implemented field carrying the quantity of DLT financial instruments acquired or disposed of. Under the current RTS 22 transaction reporting schema, such information is reported in Field 30 (Quantity).
294. Both fields provide important information under the MiFIR transaction reporting requirement. While RTS 22 Field 30 (Quantity) gives insights into the volume of financial transactions executed, RTS 22 Field 33 (Price) specifies the value of said financial transactions. Under the DLT Pilot Regime, thus, it may be sensible to be required to report similar information regarding the DLT financial instruments traded. The decision of whether such information is stored on-chain or off-chain by making use of other kinds of databases is a matter of implementation design implying a choice to be made internally by the DLT market infrastructure when building its system. As described in Paragraph 263, one potential applicant has designed their system in such a way as that information regarding price and quantity will be stored directly on Ethereum as it, in their view, constitutes the “core transaction”.
295. On the one hand, this information is valuable to the respective NCAs to better understand the amount of trading activity occurring, while also gaining a better understanding of the size of Ethereum transactions in DLT financial instruments. Furthermore, the DLT Pilot Regime places certain restrictions on issuers of DLT financial instruments. For example, if a company intends to issue shares on a DLT market infrastructure, the issuer’s market capitalisation must be less than EUR 500 million.
296. Moreover, in cases where the aggregate market value of all DLT financial instruments admitted to trading on a DLT market infrastructure has reached EUR 9 billion, the DLT market infrastructure’s operator must activate their respective transition strategy. Restrictions regarding the permitted aggregate market value also exist, as, at the moment a new DLT financial instrument is admitted to trading on a DLT market infrastructure, said aggregate market value may not exceed EUR 6 billion.¹⁰⁸ All in all, for regulators as well as DLT market infrastructure operators, it is thus crucial to know the prices of the various DLT financial instruments traded in order to ensure compliance with the outlined restrictions, in

¹⁰⁸ <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32022R0858#d1e717-1-1>

case the market prices of DLT financial instruments are the foundation for calculating the outlined threshold values.

297. As the options to design Ethereum smart contracts, and thereby DLT financial instruments traded on Ethereum, are vast, a complete overview over potentially relevant fields to be reported to a regulator is not feasible. To reach a certain level of standardisation for transaction reporting, the regulator must place restrictions on the structure of smart contracts. To come up with such restrictions, the DLTR could be leveraged to gain further insights into market practices.

3.2.4.4 Fields relevant for on-chain analysis

298. On-chain analysis is a further way of looking at Ethereum data, i.e., Ethereum transactions executed and recorded on the Ethereum blockchain to gain an enhanced understanding of on-chain activities. It can be used to analyse the number of Ethereum transactions taking place at a certain time of day or time of month, the amount of funds being transferred around the network, or which network participants are especially active. In the case of Ethereum transactions in DLT financial instruments, two native Ethereum transaction fields may be especially relevant when engaging in on-chain analysis. These two fields are *gasLimit* and *data*.

299. The *gasLimit* fields could be relevant because it is typically correlated with the size of the accompanying Ethereum transaction. As described in Paragraph 228, a *regular* Ethereum transaction typically has a *gasLimit* of 21,000 units. However, as Ethereum transactions can be more complex than solely sending funds from one address on the network to another address, the *gasLimit* set can be significantly higher. This means, a network participant could set a significantly higher *gasLimit* for an Ethereum transaction if they expect it to take up large computational resources.

300. Hence, an increase in the *gasLimit* set may point towards more complex Ethereum transactions, i.e., a larger Ethereum transaction containing more data and requiring greater network resources. This could be the case, where additional attachments are sent as part of an Ethereum transaction. Attachments to an Ethereum transaction can be any arbitrary, including malicious, data a market participant wants to include in the Ethereum transaction, e.g., insider information.

301. This leads to the other native Ethereum field relevant for on-chain analysis. Data included in an Ethereum transaction would be encoded in Ethereum's *data* field. This data can range from simple messages, to files, pictures, or other data representable by bytes. Hence, it is also possible that such data may include relevant or illicit data from a regulator's point of view.

302. In such a scenario, on-chain analysis can be a powerful tool for gaining a better understanding of activities on Ethereum. By analysing native fields such as *gasLimit* and *data*, a regulator may be able to gain better insights into the size and complexity of Ethereum transactions in DLT financial instruments. This information could prove especially relevant from a regulatory point of view, as it could be used as an entry point to further analyse suspicious, e.g., especially large, Ethereum transactions.
303. Another example of the usefulness of on-chain analysis would be to supervise which trading parties are especially active on a network and who they interact with. A regulator could monitor certain Ethereum account addresses and assess who they typically engage in Ethereum transactions with or which DLT financial instruments they trade frequently. This could be done by supervising *to* and *from* addresses on the network.
304. Excessive purchasing of a specific DLT financial instrument could be a sign of trying to artificially inflate the price of such instrument and therefore constitutes market manipulation. While information regarding trading parties is proposed to be transmitted to regulators anyways, on-chain analysis provides regulators with the ability to monitor market activities in real-time. It would also allow to assess the correctness of submitted transaction reports.
305. Furthermore, as described in Paragraph 220, on-chain analysis can prove useful as events emitted by smart contracts can be “listened” to and further analysed, e.g., regarding their overall frequency or timing. For example, if a DLT market infrastructure offers administrative functions regarding the correction of Ethereum transactions in DLT financial instruments, they could design the respective smart contracts in such a way, that every correction will lead to the emitting of an event *Correction*.
306. On-chain analysis could then be used to, for instance, identify how often such corrections occur and who is involved in them. Such analyses would provide insights into the types of Ethereum transactions being executed. Further, it may be helpful in assessing the robustness and quality of entities’ reporting systems.¹⁰⁹

3.2.5 Conclusion

307. In conclusion, Ethereum transactions do provide for more native fields that could be likened to the current RTS 22 transaction reporting schema when compared with Corda and Hyperledger Fabric. While these fields are a starting point for the analysis and understanding of an Ethereum transaction, they may not be sufficient for a regulator to fulfil

¹⁰⁹ In order to gain a comprehensive understanding of fields that are relevant for on-chain analysis on Ethereum, Section 3.3.4. of this study should be read in conjunction with Section 3.5.2 "Additionally relevant fields to perform on-chain analysis" of the Study on extraction of transaction data [\[REF\]](#)

their mandates regarding the supervision of trading activity. Therefore, additional information should be included in transaction reports regarding Ethereum transactions in DLT financial instruments.

308. Besides the fields outlined in Table 12, the integration of a DTI would provide further information regarding the DLT financial instruments traded, as well as the DLTs, on which the trading occurs. Similarly, as a DTI can also be assigned to an e-money token, its reporting could be useful in that it explains the e-money token's technical attributes in more detail. Moreover, reporting on the smart contract address may also be sensible.

309. Overall, it may be a good idea to see how DLT market infrastructures design and implement their envisioned architectures. The DLTR hence could be used to gain visibility on common market practices and lay an appropriate foundation for further standardisation and harmonisation from a regulatory standpoint.

3.3 Hyperledger Fabric

3.3.1 Background

310. Hyperledger Fabric is a private and permissioned DLT platform, which is open source and was established in 2015 under the Linux Foundation. Its modular and configurable architecture allows for a variety of use cases, including but not limited to banking and financial services. Hyperledger Fabric also finds application in use cases pertaining to the representation of supply chains.

311. Hyperledger Fabric has a two-component ledger system comprised of the world state and the transaction log, i.e., the blockchain. While the world state can be considered the ledger's database as it describes its state at any given point in time, the blockchain keeps track of all Hyperledger transactions, which have resulted in the state of the ledger as it currently exists.¹¹⁰ This means, the blockchain determines the world state. The relationship between the components is outlined in Figure 18 below.

¹¹⁰ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/blockchain.html#what-is-hyperledger-fabric>

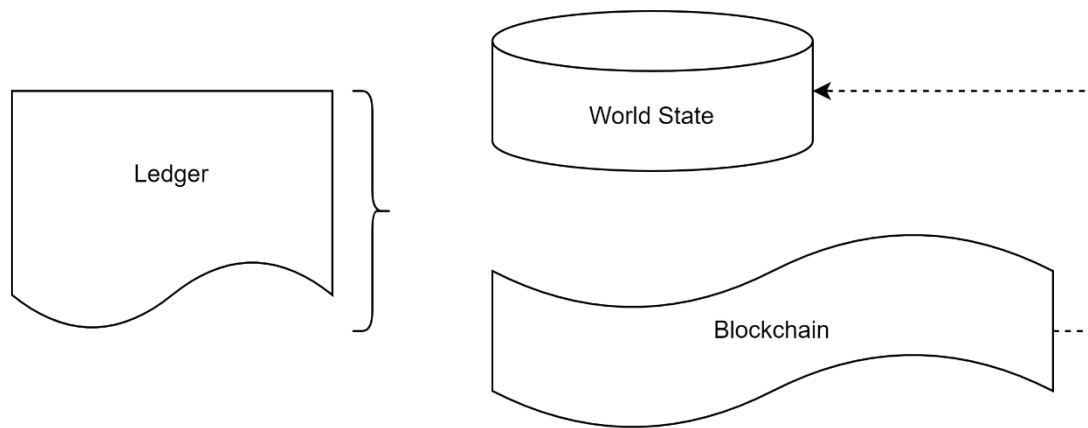


FIGURE 18: HYPERLEDGER FABRIC'S LEDGER COMPONENTS

312. The world state holds the current state of assets of a network. It can be queried to better understand the types of assets currently stored on it. In the context of the DLT Pilot Regime, the world state of a Hyperledger Fabric network could be queried to understand the types of DLT financial instruments traded on it or to see, which network participant owns which DLT financial instruments. The world state is implemented as a database and different options exist depending on the type of data to be stored in it.

313. Popular options for this database are LevelDB and CouchDB.¹¹¹ If a market participant submits a Hyperledger Fabric transaction to the network, it must first be signed as defined within the endorsement policy referred to in Paragraph 320 before it updates the world state and is stored in the applicable database.¹¹² In order to retrieve ledger state information, the world state database can be easily queried.¹¹³

314. The blockchain is a sequence of interlinked blocks and each of its blocks contains a sequence of Hyperledger Fabric transactions. Overall, the blockchain contains historical records of how assets on the network change. In doing so, the blockchain determines the world state, as mentioned in Paragraph 311. Each of the block headers making up the blockchain itself contains a hash of the block's Hyperledger Fabric transactions and a hash of the prior block's header. This design links ensures network and data security.¹¹⁴

315. Members of the Hyperledger Fabric network must enrol through a trusted Membership Service Provider (MSP).¹¹⁵ Hyperledger Fabric's default MSP uses X.509v3 certificates as

¹¹¹ <https://hyperledger-fabric.readthedocs.io/en/latest/ledger.html>

¹¹² <https://hyperledger-fabric.readthedocs.io/en/release-2.5/ledger/ledger.html#world-state>

¹¹³ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/ledger/ledger.html#world-state-database-options>

¹¹⁴ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/ledger/ledger.html#blockchain>

¹¹⁵ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/blockchain.html>

identities.¹¹⁶ These identities are crucial as they determine which accesses and permissions network participants have. X.509v3 certificates are then issued and managed by a Certificate Authority (CA).¹¹⁷ The issued X.509v3 certificates contain the CA's digital signature and link the network participant to the network participant's public key. Hence, the certificate is used to identify network participants whenever they interact with the network.

316. A Hyperledger Fabric network can also consist of multiple CAs, depending on network configuration. Multiple CAs may improve efficiency due to a distribution of the overall workload. Typically, the CA is operated and maintained by a trusted third party. This could be a government agency, a regulator, or as part of the DLT Pilot Regime, a DLT market infrastructure.
317. A further element of Hyperledger Fabric are policies. Policies are used to define decision making on the network. In doing so, they outline the rights a network participant has.¹¹⁸ For instance, policies prescribe the accesses network participants have or the amount of network participants that must be in agreement regarding updates to a channel.
318. Channels are useful in order to engage in private and confidential Hyperledger Fabric transactions.¹¹⁹ All Hyperledger Fabric transactions occur on channels and each channel contains its own Hyperledger Fabric transaction ledger.¹²⁰ Hyperledger Fabric transactions in general are explored further in 3.3.3. Typically, various organisations come together to form a channel on Hyperledger Fabric.¹²¹
319. Smart contracts on Hyperledger Fabric are called chaincodes. Chaincodes are used to handle the business logic previously defined by the network participants.¹²² Hence, chaincodes, among other things, are involved in the execution of Hyperledger Fabric transactions. They, for instance, assess whether a certain Hyperledger Fabric transaction is valid.
320. Chaincodes further always possess a so-called endorsement policy. Endorsement policies define which network participants within the respective channel must run the chaincode and endorse the execution results in order for Hyperledger Fabric transactions to be considered valid. To validate the Hyperledger Fabric transaction, the validating peers

¹¹⁶ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/identity/identity.html>

¹¹⁷ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/identity/identity.html#certificate-authorities>

¹¹⁸ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/policies/policies.html#what-is-a-policy>

¹¹⁹ <https://hyperledger-fabric.readthedocs.io/en/latest/channels.html>

¹²⁰ <https://docs.aws.amazon.com/managed-blockchain/latest/hyperledger-fabric-dev/hyperledger-work-with-channels.html>

¹²¹ <https://hyperledger-fabric.readthedocs.io/en/release-2.4/network/network.html#what-is-a-blockchain-network>

¹²² <https://hyperledger-fabric.readthedocs.io/en/release-1.3/chaincode.html>

ensure that it contains a sufficient number of endorsements from the expected sources, while also checking the validity of these sources.¹²³

321. Hyperledger Fabric provides Software Development Kits (SDKs), to enable clients, i.e., end users to interact with the underlying chaincodes and blockchain protocol. The SDKs are provided in three programming languages, which are Java, Golang, and Node.js. They enable clients to execute functions to initiate Hyperledger Fabric transactions, retrieve historical Hyperledger Fabric transaction data, or query information from the underlying ledger. This can be done to retrieve block data of a specific numbered block.
322. Another crucial part of Hyperledger Fabric networks are peers or peer nodes. Peers are in charge of managing ledgers and chaincodes as well as transaction proposals and endorsements.¹²⁴ There are various types of nodes in Hyperledger Fabric that differ in their responsibilities.
323. So-called endorsers, i.e., endorsement nodes are responsible for verifying and approving Hyperledger Fabric transactions. To do so, they, among other things, run the associated chaincode. The number of peers which must endorse a specific Hyperledger Fabric transaction depends on the applicable endorsement policy.¹²⁵
324. So-called orderers, i.e., ordering nodes, are responsible for collecting relevant information regarding Hyperledger Fabric transactions from the endorser nodes. They take care of arranging submitted Hyperledger Fabric transactions into a well-defined sequence and package them into the blocks making up the blockchain.¹²⁶ Similar to CAs, multiple orderers can be part of a Hyperledger Fabric network depending on network specifications and requirements. Orderers do not see the contents, i.e., the data contained within the Hyperledger Fabric transactions, as they merely order them chronologically rather than open them.
325. Furthermore, in cases where one wants to bypass the orderers altogether, this can be achieved multiple ways. For instance, Hyperledger Fabric's private data feature can be utilised. Data, which is stored in private collections, is shared only between network participants that are permitted to view said data.¹²⁷ This feature is used to keep certain Hyperledger Fabric transactions and their associated data secret from other network or even channel participants.

¹²³ <https://hyperledger-fabric.readthedocs.io/en/latest/endorsement-policies.html>

¹²⁴ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/peers/peers.html>

¹²⁵ <https://hyperledger-fabric.readthedocs.io/en/release-2.0/Fabric-FAQ.html>

¹²⁶ https://hyperledger-fabric.readthedocs.io/en/release-2.2/orderer/ordering_service.html#:~:text=Ordering%20service%20nodes%20receive%20transactions,and%20package%20them%20into%20blocks.

¹²⁷ <https://stackoverflow.com/questions/74313642/hyperledger-fabric-transaction-payload-visible-in-ordering-node>

326. Consensus on Hyperledger Fabric is reached by following three distinct steps, endorsement, ordering, and validation. The endorsement step is driven by the respective endorsement policy upon which a Hyperledger Fabric transaction is endorsed by other network participants. Secondly, within ordering, the endorsed Hyperledger Fabric transactions are ordered as agreed upon and in the way in which the eventually will be committed to the ledger. Lastly, validation refers to taking a block containing ordered Hyperledger Fabric transactions and validating their correctness, by, among other things, checking the applicable endorsement policy and assessing whether double spending has occurred.¹²⁸

3.3.2 Applied methodology

327. The methodology applied to study Hyperledger Fabric and Hyperledger Fabric transactions followed a purely theoretical approach. To conduct the analysis, Hyperledger Fabric's official documentation was utilised to explain the technology's main components along with its possible configurations and use cases.¹²⁹

3.3.3 Main elements of a transaction

3.3.3.1 Definition of a Hyperledger Fabric transaction

328. A Hyperledger Fabric transaction is defined as a request to update the shared ledger. In doing so, it captures changes to the world state defined in Paragraph 312.¹³⁰ Hyperledger Fabric transactions include one or more operations such as adding or modifying data. They always result in a set of key-value pairs.¹³¹ Key-value pairs represent assets, i.e., objects on the network and are also called ledger states. They can range from DLT financial instruments to legal documents or real estate.¹³² The Hyperledger Fabric transaction flow is depicted in Figure 19 below.

329. As outlined in Figure 19, multiple steps and parties are involved in a Hyperledger Fabric transaction. The process starts with the sending of a Hyperledger Fabric transaction proposal to a sufficient number of peers (*Step 1*). The exact number of peers the Hyperledger Fabric transaction needs to be sent to is detailed in the applicable endorsement policy.

¹²⁸ https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

¹²⁹ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/index.html>

¹³⁰ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/ledger/ledger.html#transactions>

¹³¹ https://hyperledger-fabric.readthedocs.io/en/release-2.2/fabric_model.html#chaincode

¹³² https://hyperledger-fabric.readthedocs.io/en/release-2.2/fabric_model.html#assets

330. The proposal itself is a request to invoke a certain chaincode function with the intent of either updating or reading the ledger.¹³³ The proposal results in the client originally initiating the Hyperledger Fabric transaction obtaining a read-write set and the relevant endorsements for the Hyperledger Fabric transaction (*Step 2*). The read set consists of the key-value pair of the object of the Hyperledger Fabric transaction as it currently exists and is read from the blockchain, while the write set outlines how the key-value pair will change upon the successful execution of the Hyperledger fabric transaction.
331. For example, if Party A intends to acquire a DLT financial instrument identifiable by the ISIN FR1234567890 and DTI 12XZ389TZ from Party B, the created read set will reflect the current state of the ledger. Hence, it may contain the information that Party B possesses the DLT financial instrument in question. The write set, on the other hand, outlines how the ledger will change upon successful execution of the Hyperledger Fabric transaction. This exemplary write set would specify that Party A is now in possession of the DLT financial instrument with ISIN FR1234567890 and DTI 12XZ389TZ. At this point, however, the ledger has not yet been updated.
332. Upon receipt of the proposal response (*Step 3*), the client that originally initiated the Hyperledger Fabric transaction broadcasts the received proposal response to the ordering service (*Step 4*). The ordering service creates a block of Hyperledger Fabric transactions and delivers it back to the peers (*Steps 5 and 6*). The receiving peers now perform two checks for all Hyperledger Fabric transactions contained within the block. Firstly, they assess whether enough endorsements were collected as prescribed by the endorsement policy. Secondly, they assess whether the Hyperledger Fabric transactions are serialisable.
333. This means, they assess whether the Hyperledger Fabric transactions can be executed in a sequential manner without causing inconsistencies. For example, if Party A executes a Hyperledger Fabric transaction and purchases the DLT financial instrument mentioned in Paragraph 331 from Party B in exchange for 100 EUR, and Party B takes the money to purchase a DLT financial instrument from Party C, the Hyperledger Fabric transaction between Party B and Party C cannot be included in a block before the Hyperledger Fabric transaction between Party A and Party B.
334. Once the peers have successfully executed the two checks, the block is validated. This means, the Hyperledger Fabric transactions themselves are also valid and can be committed to the network (*Step 7*).¹³⁴ As a last step, hence, each peer appends the newly created block to the blockchain of the respective channel and the valid Hyperledger Fabric transactions are committed to the current state of the database. The client application will

¹³³ <https://hyperledger-fabric.readthedocs.io/en/release-2.2/txflow.html>

¹³⁴ <https://wiki.hyperledger.org/pages/viewpage.action?pageId=29035620>

be notified of the successful appending of the Hyperledger Fabric transaction to the blockchain.¹³⁵

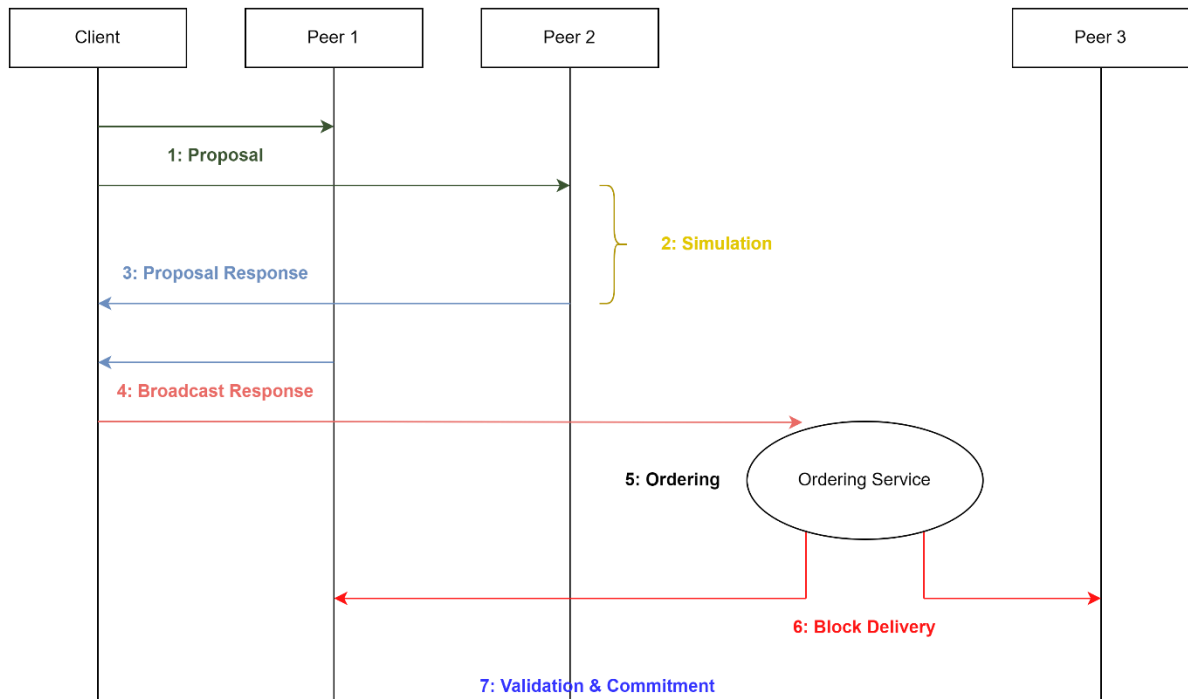


FIGURE 19: HYPERLEDGER FABRIC TRANSACTION FLOW

3.3.3.2 Types of Hyperledger Fabric transactions

335. Hyperledger Fabric transactions come in two types. The first type are *deploy* Hyperledger Fabric transactions which are used to deploy new chaincodes on the network.¹³⁶ This means, *deploy* Hyperledger Fabric transactions install and instantiate chaincodes on the network, for instance to facilitate the trading of DLT financial instruments. Typically, these chaincodes contain sets of business logic, e.g., the boundaries such a DLT financial instrument transaction must follow. Once a chaincode has been deployed to the network, it can be interacted with by making use of further Hyperledger Fabric transactions.

336. The second type are *invoke* Hyperledger Fabric transactions. This kind of Hyperledger Fabric transaction interacts with previously deployed chaincodes and their respective

¹³⁵ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/txflow.html>

¹³⁶ https://hyperledger-fabric.readthedocs.io/en/release-2.2/deploy_chaincode.html

implemented functions.¹³⁷ Hence, if a peer on a network wants to trade DLT financial instruments, they would make use of an *invoke* Hyperledger Fabric transaction.

337. The submitted *invoke* Hyperledger Fabric transaction would contain the relevant trade details, such as the quantity of the DLT financial instruments to be acquired. The previously deployed chaincode would ensure the validity of the *invoke* Hyperledger Fabric transaction by checking that the buying party has sufficient funds and the selling party possesses the DLT financial instrument in question. Upon the success of these checks, the chaincode updates the ledgers of the transacting parties accordingly to reflect the transfer of the DLT financial instrument.

338. Unlike Ethereum, Hyperledger Fabric transactions do not provide a transaction receipt. Rather, relevant Hyperledger Fabric transaction data is stored in queryable databases as described in Paragraph 312.

3.3.3.3 Established trading processes and DLT trading processes

339. Again, the trading process described in 3.1.3.3 also holds true for Hyperledger Fabric and outlines amended trading and settlement processes enabled by the technology. What the process will look like exactly and how corresponding information will be emitted as part of the Hyperledger Fabric transaction depends on the design choice of the respective DLT market infrastructure.

3.3.3.4 Structure of a Hyperledger Fabric transaction

340. Hyperledger Fabric transactions natively contain limited information as the technology allows for flexible configurations. As mentioned in Paragraph 314, Hyperledger Fabric transactions are stored in blocks which determine changes to the overall world state. Within a block, various Hyperledger Fabric transactions are contained. The various Hyperledger Fabric transactions themselves also contain further details as can be seen in Figure 20 below.

¹³⁷ <https://hyperledger-fabric.readthedocs.io/en/release-1.3/arch-deep-dive.html#transactions>

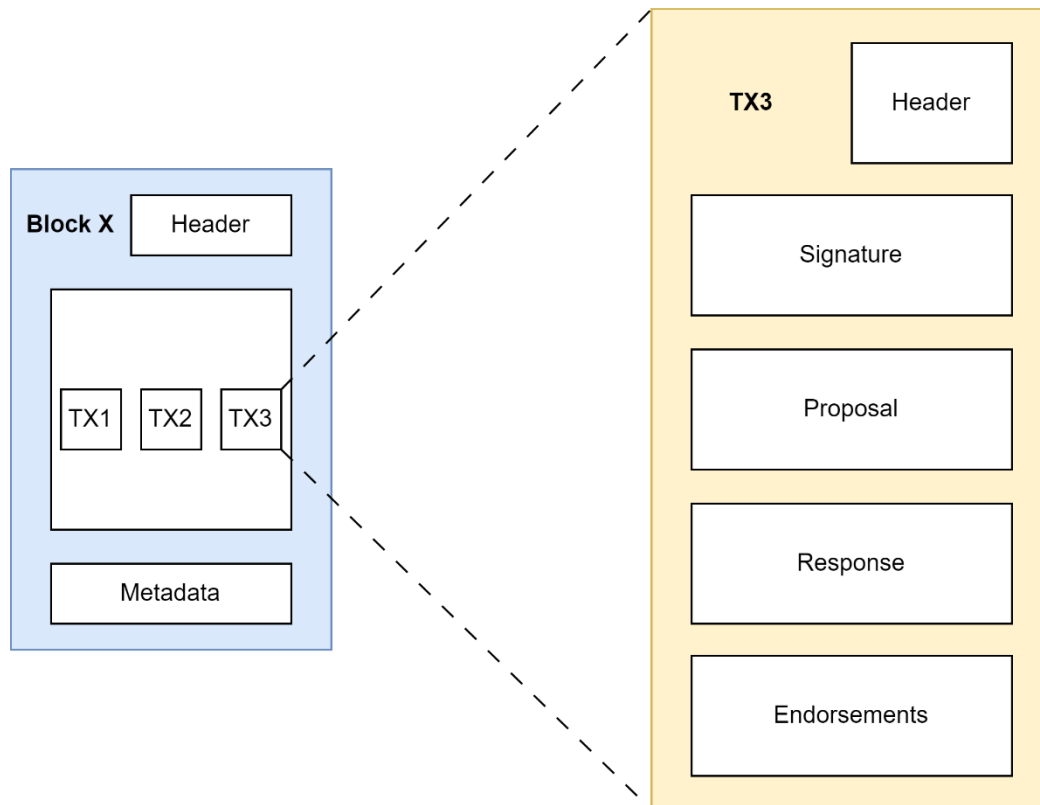


FIGURE 20: RELATIONSHIP BETWEEN BLOCK AND HYPERLEDGER FABRIC TRANSACTION

341. A block on Hyperledger Fabric contains certain information regarding its contents and computation methods. The header contains, its number, i.e., its position within the blockchain, its hash, and the hash of the previous block header. This information ensures that all blocks within a blockchain are linked to one another and make up an immutable and ordered chain of Hyperledger Fabric transactions.

342. Furthermore, the block contains an ordered list of Hyperledger Fabric transactions contained within it. The ordering service, which is defined in Paragraph 324, compiles this list during the block's creation. The block metadata contains both the certificate as well as the signature of the block creator, which used by the network nodes to verify the validity of the block.¹³⁸

343. The Hyperledger Fabric transaction itself also contains further components. Again, these are outlined in Figure 20. In the case of a Hyperledger Fabric transaction, relevant

¹³⁸ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/ledger/ledger.html#blocks>

metadata is contained within the header. Among other things, the relevant metadata includes the name of the applicable chaincode along with the chaincode's version and the *transaction ID*. The signature is also a component of a Hyperledger Fabric transaction. Every Hyperledger Fabric transaction must contain the signature of the network participant that initiated it.

344. The proposal of a Hyperledger Fabric transaction encodes its input parameters. This means, the proposal contains the details of the Hyperledger Fabric transaction to be executed. Among other things, the key-value pair will be included within the proposal. Generally, the details included as part of the proposal are supplied by the network participants submitting the Hyperledger Fabric transaction.

345. The response, moreover, depends on the proposal as it provides information regarding the values of the world state before and after the execution of the Hyperledger Fabric transaction. This is done via the read-write set defined in Paragraph 330. If the Hyperledger Fabric transaction is successfully validated, its intended updates, e.g., a purchase or sale of a DLT financial instrument will be executed and recorded in the ledger.

346. Lastly, a Hyperledger Fabric transaction will contain a list of signed responses from the network participants which, per implemented endorsement policy, must sign the Hyperledger Fabric transaction. Endorsement policies can be structured in a multitude of ways. For instance, they can prescribe that all network participants must endorse all Hyperledger Fabric transactions. On the other hand, they could also specify that only a subset, or a single predefined network participant, must sign the Hyperledger Fabric transaction.

a) Transaction identifiers

347. Hyperledger Fabric transactions are uniquely identifiable by their *transaction ID*. The *transaction ID* is generated by the client application submitting the Hyperledger Fabric transaction to the network. There is no prescribed method of how this *transaction ID* must be generated, however, its global uniqueness must be ensured. The Hyperledger Fabric SDKs provide for a built-in mechanism to create the *transaction ID*. The *transaction ID* further allows for additional querying of information regarding the conducted Hyperledger Fabric transaction. Its importance is further outlined in 3.3.4.3.

348. Multiple hashing algorithms can be used to create the *transaction ID* for the Hyperledger Fabric transaction. A common hashing algorithm used on Hyperledger Fabric

is SHA-256.¹³⁹ The SHA-256 hash is a secure algorithm producing a fixed-length, 256-bit hash value.

349. **Example:** If two network participants engage in a Hyperledger Fabric transaction, it will be uniquely identifiable by its accompanying *transaction ID*. In the hypothetical example of Party A selling a DLT financial instrument to Party B, a *transaction ID* could look as follows: d4f7c9029eef25bf3b4200fa1093e6c3aa6e51c6f12d6c8bb7dfc9e21aa2c89a. The produced *transaction ID* can then be used as a starting point to further explore the conducted Hyperledger Fabric transaction, as explained in 3.3.4.3.

b) Party identifiers

350. As mentioned in Paragraph 312, network participants on Hyperledger Fabric are identifiable by their respective X.509v3 certificates. These certificates allow for the encoding of a participant's identifying details.¹⁴⁰ As mentioned in Paragraph 90, X.509v3 certificates are configured in a way that they can be extended to include further relevant data. Thus, they can be used to extract relevant information from a regulatory point of view. Tampering of the details contained within the certificate will cause the certificate's invalidation.

351. **Example:** As mentioned in Footnote 10, a standard X.509 certificate, among other things, contains information regarding its issuer and validity period. Due to the configurable nature of the X.509v3 certificates, this basic information can be extended upon to include further relevant information regarding the network participants.

352. For instance, an X.509v3 certificate could be configured in such a way that it contains more detailed information regarding its holder. The certificate could be extended to include other kinds of personal data, including its holder's country of residence, their occupation, or date of birth. The contents and details of these certificates can be viewed by making use of various tools, such as OpenSSL.

c) Object identifiers

353. Assets, i.e., objects in Hyperledger Fabric transactions are identified by a key-value pair.¹⁴¹ Key-value pairs are stored in the world state database explained in Paragraph 312. The key distinctly represents a unique identifier for the object to be transacted. It generally is a string data type, while the value type does not need to come in the form of a specific

¹³⁹ <https://stackoverflow.com/questions/59960528/hashing-algorithm-and-library-used-in-hyperledger-fabric#:~:text=Yes%2C%20Fabric%20uses%20SHA256%20for%20hashing.>

¹⁴⁰ <https://hyperledger-fabric.readthedocs.io/en/release-2.5/identity/identity.html#digital-certificates>

¹⁴¹ https://hyperledger-fabric.readthedocs.io/en/release-2.2/fabric_model.html#assets

data type. The value part of the key-value pair typically contains further data associated with the object, which can describe the object further.

354. A potential design of a key-value pair is shown in Table 13. As mentioned in previous chapters regarding Corda and Ethereum, the integration of a DTI alongside the ISIN poses various benefits, including an enhanced understanding of the token itself as well as the underlying DLT.

355. **Example:** In the context of the DLT Pilot Regime, a key-value pair for a hypothetical DLT financial instrument could consist of a key “BOND_FR1234567890”. The key in this scenario provides information regarding the type of DLT financial instrument, in this case “BOND”, while the also providing the ISIN of the hypothetical bond. If desired, this key could also be specified to include the DLT financial instrument’s DTI rather than its ISIN or both.

356. The value part of the key-value pair provides additional information further detailing the DLT financial instrument’s characteristics. In the above scenario of the DLT financial instrument being a bond, the accompanying values could specify the bond’s issuer, instrument identification code, maturity date, DTI, and other relevant information. Making use of the key-value pair allows a third party, i.e., a regulator, to understand the object being transacted on the Hyperledger Fabric network along with the object’s characteristics. Table 13 below shows a possible specification of a key-value pair for a hypothetical DLT financial instrument.

357. It is important to note that the key-value pair, once deployed to the Hyperledger Fabric network, is immutable. This means, the encoded data cannot be modified or deleted. If some of the data is erroneous or changes, a new key-value pair can be deployed to the network to reflect the correct information.

Key = BOND_FR1234567890	
Value	
Issuer	Hypothetical Corp.
Instrument identification code	FR1234567890
Maturity date	2024-31-12
DTI	12XZ389TZ

TABLE 13: KEY-VALUE PAIR FOR HYPOTHETICAL DLT FINANCIAL INSTRUMENT "BOND_FR1234567890"

d) Price, quantity, and conversion mechanisms

358. Like Corda, Hyperledger Fabric does not provide for a native currency in which Hyperledger Fabric transactions can be conducted. Therefore, it could be sensible to include a third-party payment provider in Hyperledger Fabric networks to engage in Hyperledger Fabric transactions in DLT financial instruments. Third-party payment providers can be integrated into a Hyperledger Fabric network by making use of APIs. A further possibility would be for DLT market infrastructures to develop their own e-money tokens to be used in Hyperledger Fabric transactions.¹⁴² As with the other DLTs studied, an e-money token on Hyperledger Fabric could also be assigned a DTI.
359. There are different options to specify the price and quantity of DLT financial instruments. The concrete implementation depends on the DLT market infrastructures' preferences and architecture design. Examples are provided in Paragraphs 361 – 365.
360. In terms of conversion mechanisms, there is no natively implemented function providing a conversion mechanism between e-money tokens and ISO currency on a Hyperledger Fabric network.
361. **Example:** Generally, there are various ways in which the price for a DLT financial instrument traded on Hyperledger Fabric could be specified. For instance, the price could be set by specifying it as a fixed value within the DLT financial instrument's key-value pair. It could be specified that a certain DLT financial instrument will always have a fixed price of 100 EUR. However, such a specification would not be compliant with the way a DLT market infrastructure is supposed to operate.
362. One suitable approach for the recording of the price of DLT financial instruments would be for the DLT market infrastructure to establish its price by matching buying and selling interests from network participants. This could be done by calculating the price within the smart contract, i.e., the applicable chaincode. Input parameters to accurately determine the price of a DLT financial instrument could be supply and demand on the network.
363. Another approach would be to use an API and feed accurate price points regarding the DLT financial instrument into the Hyperledger Fabric network via an external system or data source. This would be a feasible design choice in case the price of a DLT financial instrument is recorded outside of the Hyperledger Fabric network, e.g., a trading venue.
364. The quantity in a Hyperledger Fabric transaction is dependent on the DLT market infrastructures' implementation approach. One approach would be to specify a quantity of

¹⁴²

<https://hyperledger-fabric.readthedocs.io/it/latest/Fabric-FAQ.html#:~:text=Does%20the%20Hyperledger%20Fabric%20have,own%20native%20currency%20with%20chaincode.>

DLT financial instruments to be acquired or disposed of as part of the Hyperledger Fabric transaction payload, i.e., as part of the general data making up the Hyperledger Fabric transaction.

365. Following this approach, DLT market infrastructures could require for network participants to specify the precise quantity of DLT financial instruments they intend to acquire or dispose of before execution of Hyperledger Fabric transaction. The exact specification of quantity along with the desired price at which network participants intend to purchase or sell DLT financial instruments then also allows the DLT market infrastructure to accurately match interested buyers and sellers.

366. Lastly, in case a DLT market infrastructure on Hyperledger Fabric decides to issue an e-money token to be used for trading purposes, it could be sensible to allow for a conversion mechanism to ISO currency. Like the other assets on a Hyperledger Fabric network, the e-money token could be represented as a key-value pair and be assigned a DTI.

e) Other attributes

367. Hyperledger Fabric allows for the integration of APIs. APIs, application programming interfaces, can be defined as software intermediaries allowing applications to communicate with one another.¹⁴³ In the case of Hyperledger Fabric, they allow for the communication between Hyperledger Fabric and other external systems, e.g., payment providers, as mentioned in Paragraph 358.

368. **Example:** An API can be used to integrate market data, exchange rate data, price data, further company data, or any other desired data into Hyperledger Fabric, thereby including additional relevant attributes in Hyperledger Fabric transactions related to DLT financial instruments. If a regulator requires other data to be integrated into a Hyperledger Fabric network, such data can also be fed into it by making use of an API.

f) Storage of additional business fields

369. As with the other analysed DLTs, Hyperledger Fabric also allows for multiple methods of data storage. This means, business fields of relevance to the Hyperledger Fabric transaction in DLT financial instrument can be stored fully on-chain or partially off-chain. The exact method of how business fields will be stored depends on the business application, i.e., the DLT market infrastructure. Both approaches have distinct advantages

¹⁴³ <https://www.mulesoft.com/resources/api/what-is-an-api>

and disadvantages and may also depend upon further European regulations regarding personal data.

370. **Example:** In the hypothetical example that additional business fields are stored partially off-chain, a link between the off-chain and on-chain data should be implemented to have a holistic overview of the Hyperledger Fabric transaction. For example, personal data obtained during a market participant's KYC and onboarding process at a DLT market infrastructure could be stored in a separate, off-chain database. This off-chain data could be connected to on-chain data, such as the market participant's respective X.509v3 certificate, via making use of a unique identifier. Using the unique identifier could then connect any on-chain activities, i.e., engaging in Hyperledger Fabric transactions, to off-chain data about the market participant. Figure 10 outlines this approach. A similar process could be followed if reference data regarding the DLT financial instrument is stored off-chain.

371. On the other hand, all data pertaining to the market participants, the DLT financial instruments, and other regulatorily relevant data could also be stored completely on-chain. As mentioned in Paragraph 269, this approach may lead to having to consider further European regulations aimed at the safekeeping of personal data.

g) Correction and cancellation mechanisms

372. Paragraph 119 explains the logic behind cancellations and corrections regarding RTS 22 transaction reports. Hyperledger Fabric transactions cannot be deleted or removed, making the network immutable.¹⁴⁴ However, as mentioned in Paragraph 311, Hyperledger Fabric transaction data is stored in ledgers comprised of a blockchain and a world state. While the blockchain contains and keeps track of all Hyperledger Fabric transactions that have occurred on the network, the world state describes the state of the network at a given point in time.

373. This means, the world state contains the data currently making up the network, i.e., which network participants currently own which key-value pairs and other information. While no data stored on the blockchain can be erased, data making up the world state can be overwritten by way of new Hyperledger Fabric transactions. In case a Hyperledger Fabric transaction was sent erroneously, a new Hyperledger Fabric transaction must therefore be sent to rectify the error.

374. **Example:** As mentioned in Paragraph 373, a new Hyperledger Fabric transaction would have to be sent to rectify a previously incorrectly executed Hyperledger Fabric

¹⁴⁴ <https://stackoverflow.com/questions/55833327/is-it-possible-in-hyperledger-fabric-remove-some-transactions-from-blockchain>

transaction. In order to facilitate such an action, the respective DLT market infrastructure or the applicable chaincode may offer a function that will ensure that such a reversal can only be executed in case both of the initial trading parties, as well as the DLT market infrastructure itself, are in agreement that the Hyperledger Fabric transaction should in fact be reversed. Further, the chaincode could specify that another requirement for the reversal of a Hyperledger Fabric transaction is that the amending Hyperledger Fabric transaction must occur between the same trading parties as the initial Hyperledger Fabric transaction. The submitted transaction report for the cancelation should also clear state which initial Hyperledger Fabric transaction it refers to.

3.3.4 Gap analysis with respect to RTS 22

375. In the context of the following analysis, it is again assumed that the parties to a Hyperledger Fabric transaction have a valid identity on the network. In the context of the DLTR, this valid identity can be used to engage in Hyperledger Fabric transactions, including Hyperledger Fabric transactions in DLT financial instruments. For Hyperledger Fabric transactions in DLT financial instruments, the trading party will trade via a DLT market infrastructure, who will then be tasked with submitting applicable transaction reports. As there is a direct interaction between the trading parties and the DLT market infrastructure, as seen in Figure 6, every Hyperledger Fabric transaction in DLT financial instruments should result in the production of one transaction report.

3.3.4.1 Fields similar to RTS 22

376. Hyperledger Fabric contains limited native components, which are described in 3.3.3.4. Out of the outlined components, some of the data included in the header can in parts be likened to the TVTIC field under RTS 22. As mentioned in Paragraph 343, the header in a Hyperledger Fabric transaction contains its metadata and therefore its *transaction ID*. As the *transaction ID* must be globally unique and serves as a unique identifier of a Hyperledger Fabric transaction, it could be considered to be similar to the TVTIC field.

Hyperledger Fabric field	Contained in RTS 22?
Transaction ID	The transaction ID can be considered to be similar to RTS 22 Field 03 (Trading venue transaction identification code), however differences in format exist.

TABLE 14: HYPERLEDGER FABRIC FIELDS SIMILAR TO RTS 22

377. Within the proposal, and therefore the Hyperledger Fabric transaction’s payload, further regulatorily relevant information may be included. The included information could

pertain to the quantity of a DLT financial instrument acquired or disposed of or, more generally, reference data regarding the DLT financial instrument itself. Furthermore, it could include pricing information or information regarding the buyer and seller. However, these attributes are not standardised and depend on the Hyperledger Fabric transaction's configuration.

3.3.4.2 Fields not covered in RTS 22

378. Depending on the configuration of the Hyperledger Fabric transaction, various of the native components of a Hyperledger Fabric transaction are not covered by the current RTS 22 transaction reporting regime. These components are the proposal, signature, response, and endorsements, which may contain further potentially relevant data, as described in 3.3.3.4.

Hyperledger Fabric Components	Contained in RTS 22?
Proposal	No.
Signature	No.
Response	No.
Endorsements	No.

TABLE 15: HYPERLEDGER FABRIC COMPONENTS NOT COVERED BY RTS 22

3.3.4.3 Fields of particular importance

379. Certain fields included in a Hyperledger Fabric transaction may be of particular importance to properly supervise trading activity according to ESMA and NCA mandates. Firstly, the *transaction ID* generated as part of a Hyperledger Fabric transaction and included in the header is an important identifier and component in any Hyperledger Fabric transaction. As mentioned in Paragraph 347, there is no prescribed method of how this ID must be generated, although the DLT provides for a built-in mechanism.

Field to be added	Rationale
Transaction ID	Enables the unique pinpointing of a specific Hyperledger Fabric transaction occurring on the network. Either the current TVTIC field could be amended to accommodate for the format of a Hyperledger Fabric

	transaction ID or an entirely new field specific to the transaction ID could be added.
--	--

380. Among other things, the *transaction ID*'s importance stems from its ability to be a unique identifier of a Hyperledger Fabric transaction. Making use of the *transaction ID*, further relevant details regarding the Hyperledger Fabric transaction, such as the time it was submitted or its status, can be retrieved. As mentioned in Paragraph 312, information regarding Hyperledger Fabric transactions can be queried from the respective databases it is stored in.

381. Furthermore, obtaining information regarding the data included in the Hyperledger Fabric transaction's proposal and response also could be important from a regulatory point of view. As described in Paragraphs 344 and 345, they respectively encode the input parameters, i.e., the details of the Hyperledger Fabric transaction, as well as how the Hyperledger Fabric transaction will update the network's world state. Therefore, obtaining the relevant data included in these components would allow a regulator to better understand the changes, i.e., the Hyperledger Fabric transactions occurring in a Hyperledger Fabric network as well as identify the involved network participants.

3.3.4.4 Fields relevant for on-chain analysis

382. Hyperledger Fabric provides for the Hyperledger Fabric Explorer, which is a web application allowing for detailed analysis of data stored on the network's ledger. The Hyperledger Fabric Explorer allows for the viewing and querying of blocks, Hyperledger fabric transactions and their associated data, as well as chaincode and general network information.¹⁴⁵ On 12 May 2022, the Hyperledger Fabric Explorer was moved to EOL status, i.e., end of life status. While the tool can still be used, it is no longer being maintained and may no longer be suitable for detailed analyses of Hyperledger Fabric transactions.¹⁴⁶

383. Further tools exist that can be used for the analysis of Hyperledger Fabric networks in general. One of these tools is Splunk¹⁴⁷, which can be used to analyse blocks and Hyperledger Fabric transactions.¹⁴⁸ Furthermore, the tool also allows for the analysis of

¹⁴⁵

<https://www.hyperledger.org/use/explorer#:~:text=Hyperledger%20Explorer%20is%20a%20user,information%20stored%20in%20the%20ledger>

¹⁴⁶ <https://wiki.hyperledger.org/display/explorer/Hyperledger+Explorer>

¹⁴⁷ Other tools include but are not limited to Prometheus and Grafana.

¹⁴⁸ <https://splunkbase.splunk.com/app/4612>

Hyperledger Fabric chaincode events.¹⁴⁹ These insights enable a more detailed analysis of network activity.

384. An exemplary integration of Splunk into a Hyperledger Fabric network could provide insights into the channels being used for the execution of Hyperledger Fabric transactions and what the underlying chaincodes are.¹⁵⁰ In addition, it is possible to display the execution time, *transaction ID*, and overall status (success or failure) of a Hyperledger Fabric transaction. An exemplary screenshot of these items is provided in Figure 21 below.

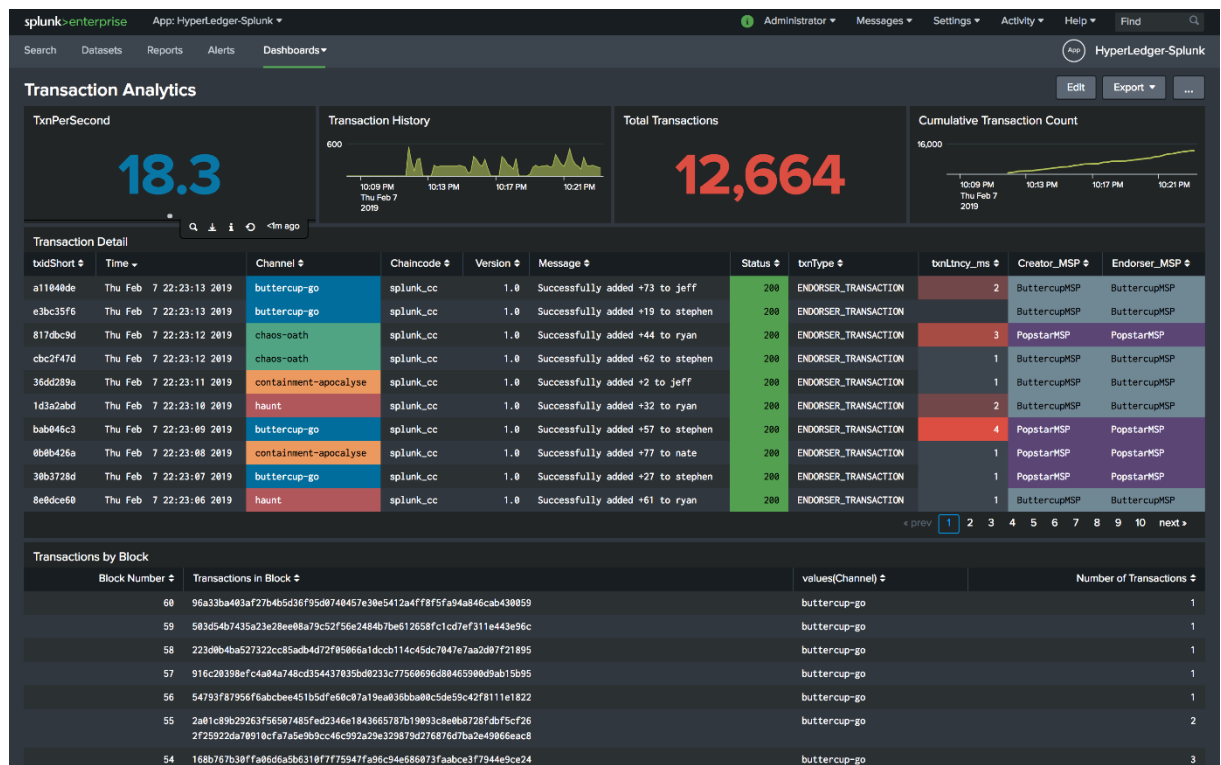


FIGURE 21: HYPERLEDGER FABRIC TRANSACTION ANALYTICS (SPLUNK EXAMPLE)

385. Furthermore, the tool also allows for more macro-level analyses regarding Hyperledger Fabric networks. For instance, it can provide insight into the number of overall Hyperledger Fabric transactions, the number of Hyperledger Fabric transactions per second, and when there may be spikes in trading volume. This allows for improved analyses regarding network usage and the overall health of the network. An exemplary screenshot of these items is provided in Figure 22 below.

¹⁴⁹ <https://www.hyperledger.org/learn/publications/splunk-sp-case-study>

¹⁵⁰ <https://splunkbase.splunk.com/app/4605>

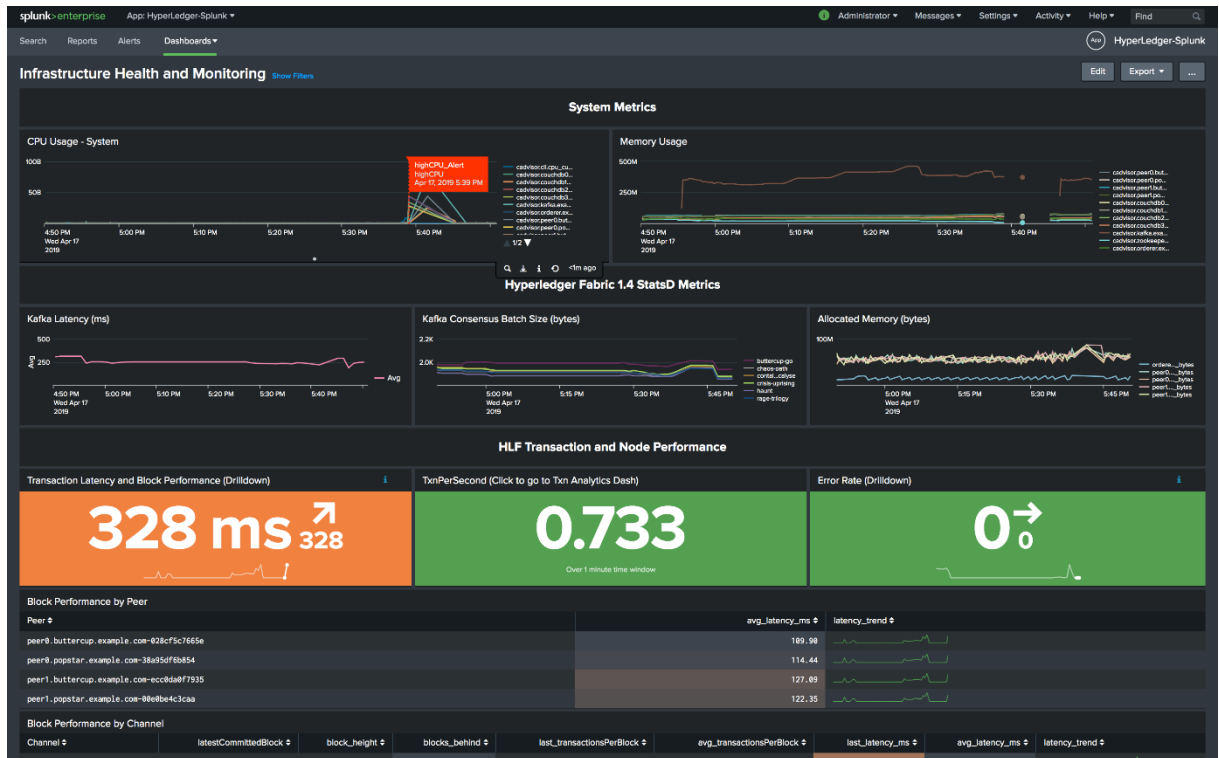


FIGURE 22: HYPERLEDGER FABRIC INFRASTRUCTURE ANALYTICS (SPLUNK EXAMPLE)

386. Splunk can also provide more insights into the setup and architecture of a Hyperledger Fabric network. This means, it can provide insights regarding, for instance, the number of peers, databases, or CAs included in the network. Therefore, such data can be helpful to understand the network's size, complexity, and security. An exemplary screenshot of these items is provided in Figure 23 below.



FIGURE 23: HYPERLEDGER FABRIC ARCHITECTURE ANALYTICS (SPLUNK EXAMPLE)

387. It is important to mention that Splunk is only one of various tools that can be used to analyse Hyperledger Fabric transactions and networks. The analysis of relevant components of a Hyperledger Fabric transaction, using one of the outlined tools, depends on the configuration of the transaction and network.

388. Making use of analytics tools can be a sensible idea to gain further insights into the inner workings of a Hyperledger Fabric network, provided that the network is configured in such a way as to contain meaningful information. Therefore, appropriate analytics tools should be designed based on the respective channel or network implementation, rather than using standardised approaches for all different kinds of Hyperledger Fabric networks.

3.3.5 Conclusion

389. Similar to Corda, Hyperledger Fabric provides for very limited native information and allows for significant flexibility in how a Hyperledger Fabric transaction can be configured and what information it can contain. Therefore, in case a regulator intends to get a comprehensive overview of network activities, clear guidelines regarding the configuration of Hyperledger Fabric transactions are necessary to ensure appropriate market supervision. These guidelines are especially relevant regarding the proper identification of

transacted objects, trading parties, as well as price and quantity of DLT financial instruments acquired or disposed of. If applicable, a general standardisation of DLT transactions as such may also be sensible.

390. Besides Hyperledger Fabric's native *transaction ID* field proposed to be added, further information may be relevant from a regulatory viewpoint. To enhance object identification, the integration of a DTI could be sensible for the same reasons as touched upon in the Corda and Ethereum chapters. This is also true in cases, in which a DLT financial instrument has been acquired by means of an e-money token that has been assigned a DTI.

391. Due to the extensive flexibility Hyperledger Fabric offers, it may be a good idea to see how DLT market infrastructures design and implement their envisioned architectures. The DLTR hence could be used to gain visibility on common market practices and lay an appropriate foundation for further standardisation and harmonisation from a regulatory standpoint.

4 Annexes

4.1 Annex I

The below description of steps refers to the setup of the private network discussed in Paragraph 23.

The minimum requirements for a testing environment with components on separate virtual machines on Corda are a 2 CPU Core with 4 GB memory. Further, the recommended production specification for components on separate virtual machines is a 4 CPU Core with 8 GB memory. For our testing purposes, a virtual environment was chosen, which, if needed, could also execute more complex tests. Therefore, a network was set up with the following specifications: A virtual machine in Microsoft Azure Standard D8s v3 with 8 virtual CPUs and 32 GB memory. The operating system used was Windows 10 and IntelliJ IDE was the used integrated development environment.

Afterwards, the prerequisites including Java SDKs, Gradle, and Git along with the Corda software including its examples in Java programming language were downloaded. Then, the Gradle script provided by Corda was executed in order to install the required dependencies and the Corda software.

As a next step, the `deployNodes` script, also provided by Corda, was executed in order to set up the network in an automated manner.

Now, each of the nodes can be interacted with via separate terminal windows. Hence, a Corda transaction flow was initiated from Party A's node. The `ExampleFlow$Initiator` is a java class, which encapsulates the business logic to be implemented. In our example, the `ExampleFlow` implements the IOU business logic such that a Party A issues an IOU to a Party B. In Corda, business logic is implemented in Flows. Flows in the context of DLT technology can be considered smart contracts. In a nutshell the `ExampleFlow$Initiator` does the following:

- Collect the required information. Here the required information is the Party and the `iouValue`
- Generate the transaction with the required information
- Verify the transaction
- Sign the transaction
- Send it over to the other party

After the transaction flow's execution, Party B's vault was queried by executing `>>> run vaultQuery contractStateType: com.example.state.IOUState`. The vault query showed that the Corda transaction with txhash `754098301241009C775E3D5A84B0DFD6F821684E69656E97CFE1F9831C3EA45C` was successful, as the vault's value was 50. This is shown in Figure 24 below.

```
Thu Dec 08 17:32:01 UTC 2022>>> run vaultQuery contractStateType: net.corda.samples.example.states.IOUState
states:
- state: !<net.corda.samples.example.states.IOUState>
  value: 50
  lender: "O=PartyA, L=London, C=GB"
  borrower: "O=PartyB, L=New York, C=US"
  linearId:
    externalId: null
    id: "5a8681dd-a56b-4b29-889b-6c9a7a7ddd88"
  contract: "net.corda.samples.example.contracts.IOUContract"
  notary: "O=Notary, L=London, C=GB"
  encumbrance: null
  constraint: !<net.corda.core.contracts.SignatureAttachmentConstraint>
    key: "a5q9DsNvGhYxYyqA9wd2eduEAZ5AXHgJTbTEw3G5d2maAq8vtLE4kZHgCs5jcB1N31cx1hpsLeqG2ngSysVHqcXhbNts6SkRwDaV7xNcr6MtcbufGUchxrnedBb6"
  ref:
    txhash: "754098301241009C775E3D5A84B0DFD6F821684E69656E97CFE1F9831C3EA45C"
    index: 0
statesMetadata:
- ref:
  txhash: "754098301241009C775E3D5A84B0DFD6F821684E69656E97CFE1F9831C3EA45C"
  index: 0
  contractStateClassName: "net.corda.samples.example.states.IOUState"
  recordedTime: "2022-12-08T17:32:01.620Z"
  consumedTime: null
  status: "UNCONSUMED"
  notary: "O=Notary, L=London, C=GB"
  lockId: null
  lockUpdateTime: null
  relevancyStatus: "RELEVANT"
  constraintInfo:
    constraint:
      key: "a5q9DsNvGhYxYyqA9wd2eduEAZ5AXHgJTbTEw3G5d2maAq8vtLE4kZHgCs5jcB1N31cx1hpsLeqG2ngSysVHqcXhbNts6SkRwDaV7xNcr6MtcbufGUchxrnedBb6"
totalStatesAvailable: -1
stateTypes: "UNCONSUMED"
otherResults: []
Thu Dec 08 17:41:57 UTC 2022>>> _
```

FIGURE 24: VAULT QUERY OF SAMPLE CORDA TRANSACTION

4.2 Annex II

The below table maps the details to be reported in transaction reports as part of Table 2 of the Annex to RTS 22 under MiFID II/MiFIR to the fields natively included in a transaction per the definition of a transaction according to the three DLTs.

Number	Field	Corda	Ethereum	Hyperledger Fabric
1	Report status	Not natively.	Not natively.	Not natively.
2	Transaction Reference Number	Not natively.	Not natively.	Not natively.
3	Trading venue transaction identification code	Corda transactions natively contain a <i>txhash</i> field, uniquely identifying any Corda transaction.	Ethereum transactions natively contain a <i>transaction hash</i> , uniquely identifying any Ethereum transaction.	Hyperledger Fabric transactions natively contain a <i>transaction ID</i> , uniquely identifying any Hyperledger Fabric transaction.
4	Executing entity identification code	Not natively.	Not natively.	Not natively.
5	Investment firm covered by Directive 2014/65/EU	Not natively.	Not natively.	Not natively.
6	Submitting entity identification code	Not natively.	Not natively.	Not natively.
7	Buyer identification code	Not natively.	Natively contains a <i>to</i> field, which is a unique identifier to specify the	Not natively.

			recipient of an Ethereum transaction.	
8	Country of the branch of the buyer	Not natively.	Not natively.	Not natively.
9	Buyer – first name(s)	Not natively.	Not natively.	Not natively.
10	Buyer – surname(s)	Not natively.	Not natively.	Not natively.
11	Buyer – date of birth	Not natively.	Not natively.	Not natively.
12	Buyer decision maker code	Not natively.	Not natively.	Not natively.
13	Buy decision maker – First Name(s)	Not natively.	Not natively.	Not natively.
14	Buy decision maker – Surname(s)	Not natively.	Not natively.	Not natively.
15	Buy decision maker – Date of birth	Not natively.	Not natively.	Not natively.
16	Seller identification code	Not natively.	Natively contains a <i>from</i> field, which is a unique identifier to specify the initiator of an	Not natively.

			Ethereum transaction.	
17	Country of the branch of the seller	Not natively.	Not natively.	Not natively.
18	Seller – first name(s)	Not natively.	Not natively.	Not natively.
19	Seller – surname(s)	Not natively.	Not natively.	Not natively.
20	Seller – date of birth	Not natively.	Not natively.	Not natively.
21	Seller decision maker code	Not natively.	Not natively.	Not natively.
22	Sell decision maker – First Name(s)	Not natively.	Not natively.	Not natively.
23	Sell decision maker – Surname(s)	Not natively.	Not natively.	Not natively.
24	Sell decision maker – Date of birth	Not natively.	Not natively.	Not natively.
25	Transmission of order indicator	Not natively.	Not natively.	Not natively.
26	Transmitting firm identification code for the buyer	Not natively.	Not natively.	Not natively.
27	Transmitting firm identification code for the seller	Not natively.	Not natively.	Not natively.

28	Trading date time	Natively contains a <i>recordedTime</i> field, specifying the time at which a Corda transaction was committed to the ledger.	Natively contains a <i>timestamp</i> , which outlines the date and time at which the Ethereum transaction was produced.	Not natively.
29	Trading capacity	Not natively.	Not natively.	Not natively.
30	Quantity	Not natively.	Not natively.	Not natively.
31	Quantity currency	Not natively.	Not natively.	Not natively.
32	Derivative notional increase/decrease	Per definition, not in scope of the DLT Pilot Regime.		
33	Price	Not natively.	Not natively.	Not natively.
34	Price Currency	Not natively.	Not natively.	Not natively.
35	Net amount	Not natively.	Not natively.	Not natively.
36	Venue	Not natively.	Not natively.	Not natively.
37	Country of the branch membership	Not natively.	Not natively.	Not natively.
38	Up-front payment	Not natively.	Not natively.	Not natively.
39	Up-front payment currency	Not natively.	Not natively.	Not natively.
40	Complex trade component id	Per definition, not in scope of the DLT Pilot Regime.		
41	Instrument identification code	Not natively.	Not natively.	Not natively.

42	Instrument full name	Per definition, not in scope of the DLT Pilot Regime.					
43	Instrument classification						
44	Notional currency 1						
45	Notional currency 2						
46	Price multiplier						
47	Underlying instrument code						
48	Underlying index name						
49	Term of the underlying index						
50	Option type						
51	Strike price						
52	Strike price currency						
53	Option exercise style						
54	Maturity date				Not natively.	Not natively.	Not natively.
55	Expiry date				Per definition, not in scope of the DLT Pilot Regime.		
56	Delivery type						
57	Investment decision within firm	Not natively.	Not natively.	Not natively.			

58	Country of the branch responsible for the person making the investment decision	Not natively.	Not natively.	Not natively.
59	Execution within firm	Not natively.	Not natively.	Not natively.
60	Country of the branch supervising the person responsible for the execution	Not natively.	Not natively.	Not natively.
61	Waiver indicator	Not natively.	Not natively.	Not natively.
62	Short selling indicator	Per definition, not in scope of the DLT Pilot Regime.		
63	OTC post-trade indicator			
64	Commodity derivative indicator			
65	Securities financing transaction indicator	Not natively.	Not natively.	Not natively.