DS Protocol -

Securitize's Digital Ownership Architecture for Complete Lifecycle Management of Digital Securities

Carlos Domingo, Shay Finkelstein, Jorge Serna

Version 1.0.0 - June 5th, 2018

SECURITIZE

The Securitize Protocol Strategy

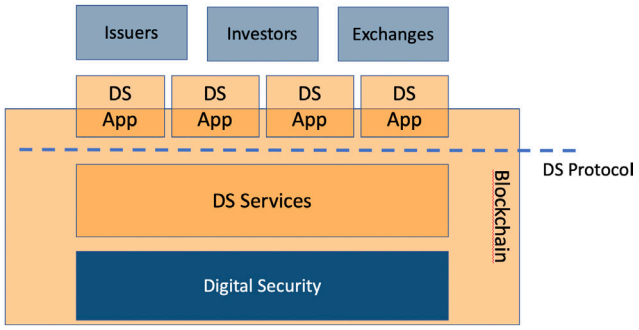
Securitize's mission is to be the leading platform for enabling compliant Digital Securities issuance and liquidity on the blockchain. To achieve this, Securitize has built a technology platform that provides the tools to issuers to manage all the elements in the full Digital Security lifecycle, including the smart contracts that actually support the token information on the Ethereum blockchain.

Securitize is designing its smart contract infrastructure by creating a flexible ecosystem that aids the main stakeholders in the Digital Security space: Issuers, Investors and Exchanges.

A Digital Ownership Architecture of DS Services and DS Apps

The potential of this ecosystem and the complexity of the different needs for each stakeholder and category of assets involved require a new architectural model. This model should allow for flexible and simple evolution by different actors. It should be a Digital Ownership Architecture that facilitates a dynamic and open participation of multiple actors.

Securitize is implementing this new architecture focused on creating a DS Services (Digital Securities Services) infrastructure which will support third party DS Apps (Digital Securities Apps) to address all aspects of the Digital Security lifecycle. We believe this ability for any actor to be able to create new DS Apps to bring more value to an existing Digital Security and extend its capabilities in an open way, is critical to ensure the future-proofing of this approach.
The interaction between the different elements is managed by DS Protocol (Digital Securities Protocol) a layered and extensible protocol. This protocol provides value to stakeholders while addressing compliance by leveraging the immutable, public, distributed ledger nature of the blockchain.



SECURITIZE'S DIGITAL OWNERSHIP ARCHITECTURE

Securitize's first implementation will work over the Ethereum blockchain, but the goal is to transfer this architectural model to additional distributed ledger systems.

Full Lifecycle Support

Other initiatives have focused in the issuance and compliance aspects for a Security Token Offering, which are very important by themselves, mostly addressing the token issuers' goals. In some cases, their platforms follow a module-based structure, which behave like plug-ins that fulfill certain functions in a narrow way on a centralized solution.

We believe the ability to openly create independent apps for any existing and future use case is a much more robust, scalable, and open solution, as proven in so many other software environments. That is why Securitize has taken it a step further with the design of the DS Protocol. The DS Protocol considers the whole lifecycle of a Digital Security-enabling applications (DS Apps) to address relevant events associated with the ownership of tokenized economic rights.

It considers not just the issuers but also investors and exchanges, and the relationships of trust that may be required among them. This is creating an ecosystem in which developers can expand the value that Digital Securities provide to stakeholders.

The DS Protocol Ecosystem

Securitize believes the Digital Securities ecosystem will be very diverse, due to the enormoussize, varied nature and specific vertical requirements of the securities market. Securitize aims to enable this diversity by using a model that will facilitate the flexible development of specific implementations for certain elements, an ecosystem of Digital Securities Apps.

The main elements in this ecosystem are:

DS Tokens:
ERC-20 compliant tokens, extended with the capabilities of the DS Protocol.

DS Apps:
Smart Contracts designed to manage specific lifecycle events for a Digital Security. Examples for this are issuance DS Apps, exchange-specific DS Apps, voting rights DS Apps or dividend issuance DS Apps.

DS Services:
The basic infrastructure of the DS Protocol, enabling lifecycle management and compliance to DS Tokens. DS Apps can access these services to fulfill their goals.

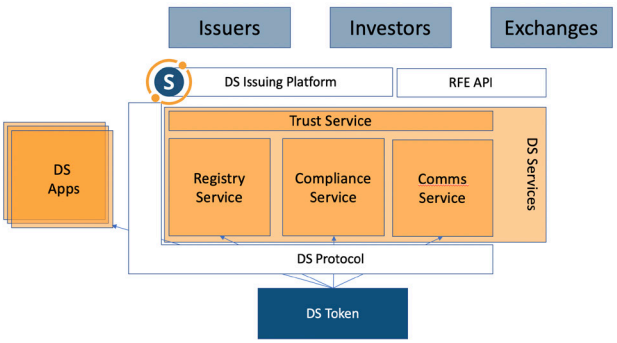
The DS Services include:

- ***Trust Service:*** *managing the relationships between the different stakeholders.*
- ***Registry Service:*** *an on-chain register of investor information.*
- **Compliance Service:** *which implements specific compliance rules applicable to a DS Token as per the Issuer requirements.*
- **Comms Service:** *enabling communication of relevant events to investors.*

To improve the investor experience, Securitize's platform extends the DS Protocol with the Ready For Exchange (RFE) off-chain API. This API will enable Exchanges to integrate into the Digital Securities ecosystem in a simple way that also improves the customer experience for investors.

For instance, it will enable the sharing of KYC information between the issuer to the authorized Exchanges, not shared on-chain due to privacy constraints. This is useful for the Issuer to be able to maintain the cap tables for investors and for exchanges to provide a contact mechanism information to the issuer (e.g., an email) so that relevant lifecycle events can be notified. Additionally, there are some REST APIs that allow access to specific
information about the investor pool (e.g., allocation of investors of a certain category according to regulation), so that they can provide the right user experience in their investordashboards.

The following figure shows a high-level representation of the different elements in the DS Protocol Ecosystem:



In the next sections, the different components in this ecosystem will be described. Subsequently, we will illustrate several user journeys that will clarify the flows involved during

The DS Token

The DS Token is an ERC-20 compliant token that also implements the additional hooks required by the DS Protocol. As per this ERC-20 specification, the DS Token contract can be queried to get the balances of tokens for specific wallets or the total supply available for the token.

To address regulatory compliance, the token contract has its transfer() and transferFrom() methods overloaded so that they check via the Compliance Service if tokens can be exchanged between specific wallet addresses.

Besides the ERC-20 methods, the DS Protocol adds methods for issuers being able to block wallets, or freeze the token due to a regulatory requirement. It also provides methods for DS Apps to be able to iterate through the investor list to execute their services, like dividend issuances.

Issuers of Digital Securities may create their own specific DS Token implementing the DS Protocol, usually through an issuance DS App or a DS Issuing Platform, like Securitize's white-label platform. The use of the DS Protocol standardizes the interaction between the token and other Smart Contracts. These Smart Contracts offer the additional securityfocused capabilities required for issuers, investors and exchanges during the Digital Security lifecycle.

DS Apps

The DS Apps are additional Smart Contracts that can be registered with a DS Token and invoked based on their own interface to trigger a specific lifecycle event. Typically, a DS App developer will provide some front-end interface to the issuer to manage the execution of this given app.

For instance, DS Tokens, whether acquired on issuance or via a secondary trading through a Security Token Exchange (STE), may convey additional rights or require some specific additional capabilities depending on their underlying asset.

Certain tokens may provide governance rights or have payback mechanisms associated with them. To this end, as part of the DS Protocol, it is possible to associate different DS Apps to a DS Token, which can provide such additional value for token holders.

The user journeys section includes specific examples of lifecycle events and how their DS Apps would work within the DS Protocol.

Trust Service

A key component in the DS Protocol is the Trust Service. The Trust Service is used by the rest of the components in the DS Protocol architecture to ensure that a given actor or DS App is allowed (in general by the issuer) to take a certain action. Relevant examples of actions controlled by the Trust Service are:

- *Using multiple Ethereum addresses in the Issuer role, but with different levels of permissions. For instance, several addresses may be allowed to add investor information to the Registry Service, but only a specific one can manage registry federation.*
- *Authorizing an Exchange to introduce additional entries into the Registry Service. Given that the model in the DS Protocol delegates gathering KYC and investor accreditation information to exchanges for new investors, only trusted Exchangeswill be authorized to do this.*
- *Authorizing a specific DS App to use some of the protocol capabilities, such as the Comms Service to be able to send communications to investors.*
- *Comms Service: enabling communication of relevant events to investors.*

To simplify the example flows in the user journeys section, the Trust Service will not be included in the diagrams. The assumption is that all actors involved have been previously authorized to do the different functions presented.

Registry Service

The Registry Service holds an on-chain register of the investors that hold a given DS Token. Each investor is identified by a unique ID, built from a hash of personal information that allows identification in a privacy-safe way, and includes the following information:

- Investor Country for regulation purposes
- Investor KYC status
- Investor Accreditation/qualification status
- Investor KYC expiry date
- Investor Accreditation expiry date
- Investor KYC information hash
- Investor Accreditation information hash

Both Issuers and Exchanges roles can interact with the Registry Service to create investors and associate investors to specific wallets.

In particular, the Registry Services will support assigning several wallets to the same investor, so that investors can manage their token allowances in a flexible way.

An additional, non-unique ID hash (for instance, based just on the investor name and birth date) may be also held in the registry to raise situations in which the same investor may have been registered twice using different identification documents. In those situations, specific methods will allow the Registry Service owner to merge investor entries into a single registry.

This case is particularly relevant when several entities are providing customer identification (for instance, the issuer and exchanges), and there are limits on the number of investors supported for a certain category (for instance, a limited number of U.S. investors), which may require avoiding double-counting of the same investor.

The Registry Service allows a federation model, by which if a match is not found for a specific investor query, it can, in turn, propagate the query to a different investor registry.

Compliance Service

The Compliance Service will handle the transfer() and transferFrom() validation calls from the DS Token contract.

It will check the sender and receiver address in the Registry Service and will enforce the constraints applicable to a given issuance. For instance, a fund issuer may want to limit its presence in the U.S. to just 99 accredited investors. Alternatively, an issuer following Reg S may require preventing flow-back from non-U.S. to U.S. investors. These rules will be enforced by the Compliance Service, allowing or rejecting specific token transfers.

To facilitate adherence to the specific requirements of an issuer for compliance, the Compliance Service will also be responsible for storing and updating data that can be required for allowing a transfer to happen.

This will be very specific to the Digital Security andthe regulatory context for issuance and trading. Examples of information that a given Compliance Service may store and update are:

- Investor-specific information, like their adherence to a subscription agreement or the hash for the signed agreement document.
- Number of investors for a certain category, for instance non-qualified investor from a certain country.
- Number of trades that have occurred in a certain period

ADDITIONAL COMPLIANCE SUPPORT

For certain classes of assets, it may be required by the applicable regulation that a token holder be able to get their tokens re-issued in case they have lost access to their previous wallet.

The Compliance Service is also responsible for allowing or preventing the issuer to allow such reissuance.

Comms Service

During the lifecycle of a security investment it is important to be able to provide information to investors. This way, token owners can be aware of liquidity events, governance events, or any other relevant aspects that are a fundamental part of a Digital Security lifecycle.

During the initial issuance through a DS Issuance Platform, information to communicate with investors (e.g. an email address) can be gathered. But this information may not be
available to the issuers for new investors accessing the tokens through exchanges in thesecondary market.

As described above, the RFE (Ready For Exchange) API may also be used by exchanges to share this information with the issuance platform; but to avoid those dependencies at the protocol level, the DS Protocol includes the Comms Service.

The Comms Service does not make any assumption on what other communication means may be available off-chain and is defined in a generic way, accepting messages addressed exclusively to Ethereum addresses. This allows the flexibility of different implementations of the Comms Service supporting different modes of integration with specific issuance platforms or investor wallets.

Access to the Comms Service functions is controlled via the Trust Service, so that it cannot be abused by non-trusted applications to send potential spam to investors.

The usage of the on-chain Comms Service also provides auditability to all the content and communications that an Issuer has shared with an Investor, that will have an immutable registry for them.

The choice to use a specific implementation for a communication service instead of relying on existing native capabilities in the Ethereum network has been based on its fitness to purpose. Whisper messaging is not being used because by design its goal is to provide dark comms, while for the Comms Service an auditable store for messages is required. Also, using Swarm for that purpose is not adequate due to its maturity level.

As new capabilities are supported and mature enough in the Ethereum network, it may be possible to have the Comms Service rely on them. For instance, it is possible that in the future the Comms Service will use the native Whisper and Swarm technologies offered by the Ethereum network as their feature-set and maturity match the Comms Service needs.

User Journeys

We will look in detail at the functionality provided by the different DS Protocol components by covering several user journeys that are part of a typical Digital Security lifecycle. The specific journeys covered are:

- **Issuance:** This is the initial process by which the Issuer gathers investors and funds and issues tokens to investors. We are assuming an issuance platform is used to deploy the DS Protocol Smart Contracts although this could be done in a number of different ways, for instance via DS Apps.

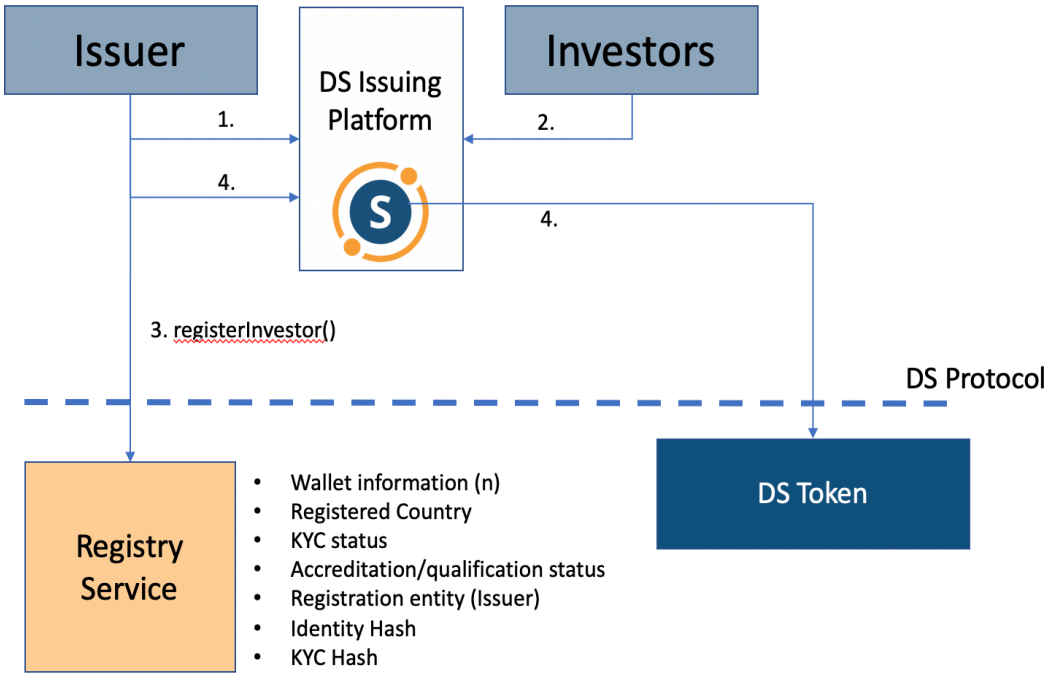
- **In-life events associated with investors' economic rights:** The specific needs for these kinds of events will be very diverse due to the multiple verticals that can issue Digital Securities. That is the reason behind the flexibility provided by the DS Protocol ecosystem approach, and its ability to support developers creating new DS Apps that will provide solutions to address these needs. To serve as an example, in this document we will consider two specific use cases, supported by DS Apps:

  – Receiving payments like dividends or buybacks.
  – Being able to exercise rights like voting in a company's decisions.

- **Trading:** Exchanges are key actors in this process, in which existing investors can sell their tokens to new investors, but in the process Issuers and Exchanges have to ensure that the process is done within the applicable regulation.

Issuance

THE FOLLOWING DIAGRAM SHOWS THE PROCESS OF A DIGITAL SECURITY ISSUANCE:



THE FOLLOWING STEPS TAKE PLACE DURING THIS PROCESS:

1. The Issuer needs to start the issuance process in order to get investors into its STO. This process will usually take place via a DS Issuing Platform or a DS App, and in the example we are considering the Securitize Platform in this context. This process is outside the scope of the DS Protocol.

2. As part of the on-boarding the issuer will identify investors, gathering KYC (Know Your Customer), AML (Anti-Money Laundering) and accreditation status for them. Investors may also have to sign a subscription agreement. Again, this happens through the DS Issuing Platform and is not part of DS Protocol.

3. The issuer must then register the information from the investors in the Registry Service. The DS Issuing Platform may manage this for the Issuer, or the Issuer can do this directly in the Ethereum blockchain via the DS Protocol method registerInvestor(). This method will provide the information required for the Digital Security lifecycle, as described in the Registry Service section above.

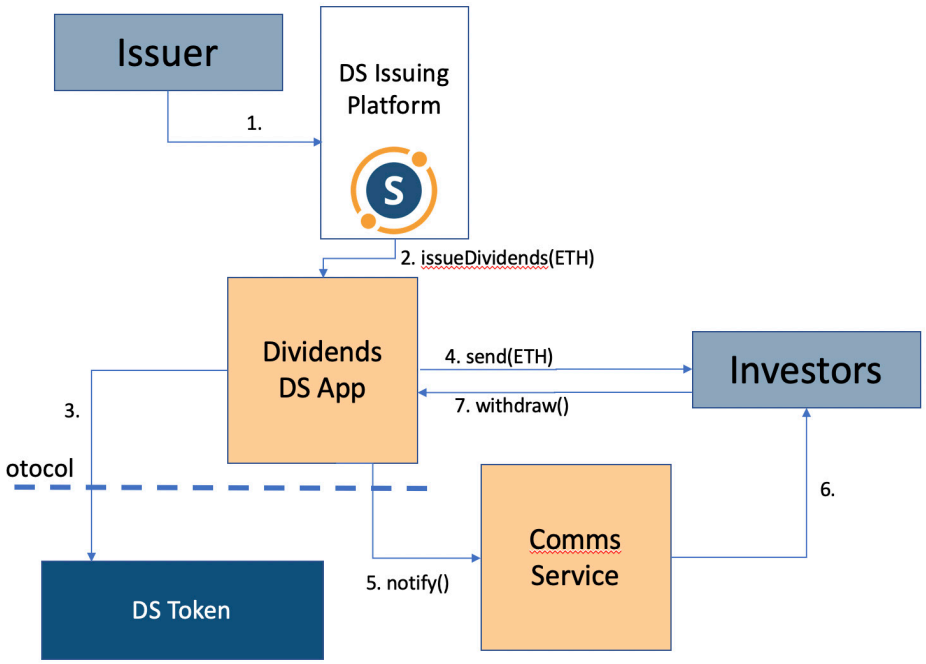
4. Once the issuance process, including collecting funds and applying issuance discounts (which is out of scope for the DS Protocol) is complete, the issuer (usually through the DS Issuing Platform) will create the DS Token contract and assign the issued tokens to the registered investors. The DS Token contract includes a reference to the Registry Service contract for that purpose.

Dividend Issuance

DS Tokens, whether acquired on issuance or via a STE, may convey additional rights from the underlying asset that require management by the DS Protocol.

Specific DS Apps can interact with a DS Token to provide these services.

THE FOLLOWING DIAGRAM PRESENTS AN EXAMPLE OF HOW A DS APP COULD IMPLEMENT THE LOGIC TO DISTRIBUTE DIVIDENDS FOR A SPECIFIC DS TOKEN.



DISTRIBUTE DIVIDENDS FOR A SPECIFIC DS TOKEN.

THE PROCESS WOULD BE:

1. An Issuer uses the console of their DS Issuing Platform to start a dividend issuance. Alternatively, depending on the specific DS App implementation, this could be triggered by an app-specific UI or directly in the blockchain without the need for a specific front-end.

2. This triggers the issueDividends() method, sending the amount of dividends to distribute to the Dividends App contract as Ether. The Dividends App is a contract following the DS App interface and DS Protocol, and has been associated with the DS Token at some point by the Issuer (potentially as part of the dividend issuance process itself). The specific logic of the contract, including the issueDividends() method or the decision to use Ether for the dividend distribution, is not part of the DS Protocol, as apps are flexible to provide any kind of function required by issuers. This is just a possible implementation for the purpose of explaining the mechanism.

3. The Dividends App then accesses the DS Token via the DS Protocol to be able to iterate through the list of investors currently active. This mechanism is specific to the DS Protocol, as ERC-20 standard does not provide a way to do this.

4. The Dividend App then sends the corresponding part of the dividends to each investor's wallet. The Dividend App may check in the Registry Service (not depicted for simplicity) the country associated with the investor, so that it can withhold the appropriate taxes when applicable. If for some reason the operation to send the Ether to a specific investor fails, their assigned dividend is kept in the Dividends App contract and an allowance is set for that investor.

5. The Dividends App uses the Comms Service to communicate to each investor the information regarding their dividend issuance, including in the cases in which the send operation has failed, so that they can retrieve the funds later.

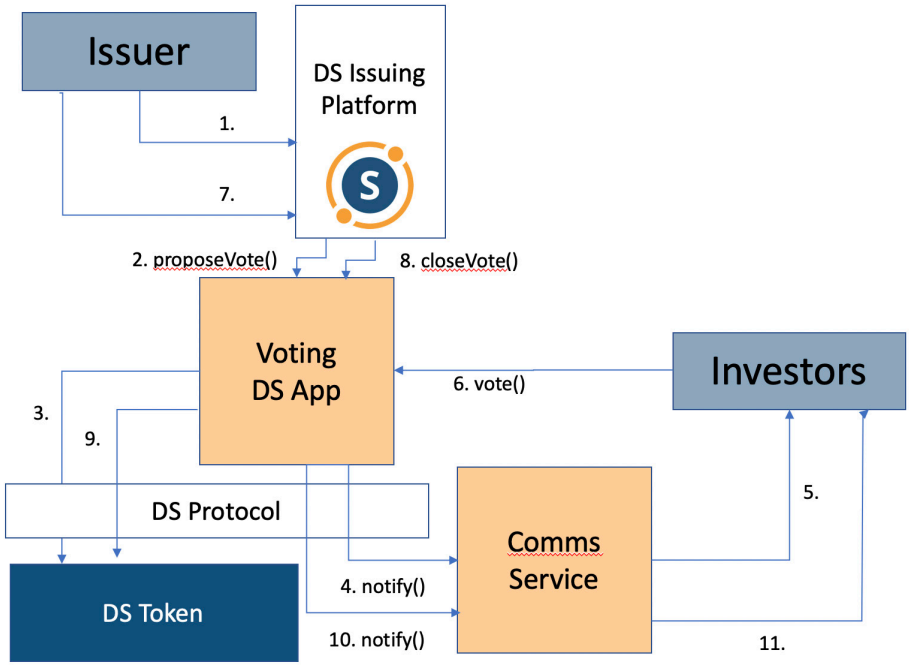
6. The Comms Service sends that communication to the corresponding investors.

7. Those investors that were not able to receive their funds in the direct method can use the withdraw() method in the contract to retrieve their funds at a later date.

Voting Process

Another example of a lifecycle event that can be managed via a DS App is the voting process in governance decisions.

A potential flow for that kind of situation is depicted next.



THIS ARE THE STEPS OF THE VOTING PROCESS

1. An Issuer uses the console of their DS Issuing Platform to start the voting event. Alternatively, depending on the specific DS App implementation, this could be triggered by an app-specific UI or directly in the blockchain without the need for a specific front-end.

2. This triggers the proposeVote() method to the Voting DS App contract. This app, as in the previous example, is a contract following the DS App interface and DS Protocol and has been associated with the DS Token by the Issuer. The definition and function of the proposeVote() method (including its name) is something decided by the smart contract developer, and not part of the DS Protocol. In this example, we assume the method (or methods) involved in its configuration sets a start and end date for the voting, plus the range of valid values to vote for.

3. The Voting DS App then accesses the DS Token via the DS Protocol to be able to iterate through the list. This is the same mechanism as presented in the previous example.

4. The Voting DS App uses the Comms Service to communicate to each investor the information regarding the voting event.

5. The Comms Service sends that communication to the corresponding investors.

6. At some point investors start issuing their votes via the vote() method into the Voting 12 DS App contract. They can do this directly through the blockchain, or the DS Issuing Platform may provide some sort of front-end to it to facilitate the interaction by investors. The Voting DS App contract only accepts votes from valid investors that produce their voting after the defined start date and before the end date of the contract.

7. Once the voting period is finished, the Issuer calls for the vote to close. Alternatively the DS Issuing Platform could trigger this event automatically once the end date for the voting has been reached.

8. This triggers the execution of the closeVote() method in the Voting DS App contract. Depending on the design of the DS App, this could be a low permission method (so that the platform could trigger it with a basic key) but would only close the voting if the end date has actually expired, or require some administration permission (e.g. only available to specific Issuer's keys) and be authoritative, closing the voting regardless of the date. These options show the flexibility that different implementations of DS Apps can bring to the Digital Securities ecosystem.

9. The Voting DS App contract goes over the list of votes collected from each investor and weighs them proportionally to the DS Tokens they hold in this specific point of time. To get the individual token balances the Voting DS App checks the DS Token (while the diagram shows the access via the DS Protocol, for this specific operation the basic ERC-20 capabilities are sufficient, which illustrates that the DS Protocol for DS Tokens is an extension of ERC-20).

10. Once all votes are considered, the Voting DS App uses the Comms Service to communicate to each investor the result of the voting event.

11. The Comms Service sends that communication to the corresponding investors.

Trading DS Tokens

Licensed Digital Securities trading platforms also known as Security Token Exchanges or STEs - will be the main actors to bring liquidity to the STO scene. The core function of STEs is to ensure fair and orderly trading of securities and the efficient dissemination of price information for those trading on that exchange.

But when a specific Digital Security is to be listed in their platform, STEs have two main concerns:

- Regulatory compliance: exchanges play a critical role in preventing non-compliant, unregistered resale of restricted securities and need to ensure this also holds true forDigital Security.
- Investor reach: listing a Digital Security is not enough if they are not able to actually sell those tokens to their own investor pool.

These are fundamentally two sides of the same problem: regulatory compliance requiresknowing specifics about the existing investment pool to limit or allow new investors into the offering.

When trading via an exchange, the trading process is affected by the mechanism applied by the exchange to interact with its users. We will consider in the following sections four approaches:

- Direct P2P Trades – an exception in which an Exchange is not involved
- A Distributed Exchange
- A Centralized Exchange providing investor-specific wallets
- A Centralized Exchange using shared wallets for trading

DIRECT P2P TRADES

Individual investors can decide to trade their tokens privately, or send them from one wallet to a different one. This mechanism may facilitate trades between existing investors that had a previous knowledge of each other. But the most usual trades will involve an Exchange of some sort.

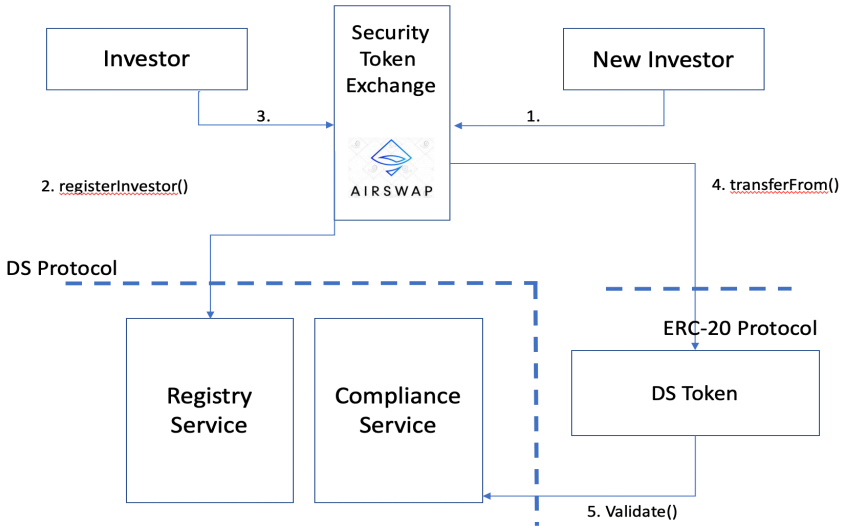
Since DS Token transfers are governed by the Compliance Service, these transfers will only be possible if they are approved by it.

In general, this will require that the originating tokens are not currently locked (for instance due to a hold period); the destination wallet is listed in the Registry Service, and the category of the receiving investor is consistent with the Compliance Service rules.

DISTRIBUTED EXCHANGE

The following diagram presents a simplified view of how trades can be executed via a STE that allows direct wallet-to-wallet trades.

Examples of this model are protocols like Swap, used by AirSwap, or the 0x protocol used by relayers like ERC dEX.



THE STEPS ARE AS FOLLOWS:

1. A new investor goes to a STE (for instance AirSwap) interested in buying a specific DS Token. As part of the on-boarding process the exchange will gather the relevant information from the customer, like KYC or accreditation status.

2. As the Issuer did for the issuance process, the Exchange will use the DS Protocol registerInvestor() method to register the details concerning the new investor into the Registry Service. The access to the registerInvestor() method will be restricted by the issuer, so that only authorized exchanges can incorporate investors to the investor pool. Exchanges will also have other alternatives for this process to improve the customer experience for the registration as part of the RFE API.

3. An existing investor who owns some DS Tokens is interested in selling, and the exchange matches both investors to do a trade. (The process of order-matching is specific for the exchange and out of the scope of the DS Protocol, but usually implies the seller authorizing an Exchange-owned Smart Contract via the ERC-20 approve() method, supported by DS Tokens, to pull tokens from its balance).

4. As part of the trade execution, the exchange (via its Smart Contract of its own) will trigger the standard ERC-20 transferFrom() method in the DS Token so that tokens are moved from the existing investor to the new one.

5. DS Tokens have overloaded the transfer() and transferFrom() methods to enable compliance in all trades. They do so by calling their associated Compliance Service contract via the DS Protocol. The Compliance Service will obtain the investors profiles from the Registry Service and validate that both have the correct categories, and are within the existing limits given the appropriate regulation so that the trade can happen. If conditions are not met, the transaction will be stopped informing via an Ethereum event of the reason why.

In this process, we have seen how the DS Protocol allows exchanges to increase the investor reach by incorporating new investors into the Registry Service, and how it ensures that only trades that are validated by the Compliance Service are permitted.

CENTRALIZED EXCHANGE – INVESTOR OWNED WALLETS

The flow in this case is very similar, but it requires the Exchange to also be creating and registering (or reusing) specific wallets for new investors, and the investors to be interacting with them.

THE FOLLOWING PROCESS WOULD HAPPEN FOR AN INVESTOR WANTING TO BUY DS TOKENS:

1. The Exchange run its own KYC process on the Investor and has their identity.
2. The Exchange creates an Ethereum wallet for the investor.
3. The Exchange registers the wallet to the investor via the Registry Service.
4. This operation may fail if the Investor is unknown, particularly in the case of a new investor. In this case:
   - The Exchange registers the investor, providing the Country/ KYC/accreditation information and requiring the investor to sign the Subscription Agreement from the issuer.
   - The Exchange registers the wallet to the investor again, this time successfully.
5. The Exchange matches the buy order with one (or several) sell order(s).
6. The Exchange transfers the tokens between wallets, using the transfer() method.
   - This will trigger the execution of the regulatory checks by the Compliance Service.
7. If no regulatory limits are surpassed the trade takes place.
8. The new investor has the DS Tokens in their Exchange wallet, and can do additional trades, or withdraw them to another wallet by registering that wallet through the exchange or issuer.

THE FOLLOWING PROCESS WOULD HAPPEN FOR AN INVESTOR WANTING TO SELL DS TOKENS

1. The Exchange has run its own KYC process on the Investor and has their identity.
2. The Exchange creates an Ethereum wallet for the investor.
3. The Exchange registers the wallet to the investor via the Registry Service.
   - This operation should be successful because the investor is preexisting, as they claim to already own the token. If this fails, the Exchange should inform the investor that there is a problem with their information.
4. The Exchange matches the buy order with one (or several) sell order(s).
   - This should work as both wallets are registered to the investor and investors can move their tokens freely across their own wallets.
5. The exchange matches the sell order with a buy order.
6. The Exchange transfers the tokens between wallets, using the transfer() method.
   - This will trigger the execution of the regulatory checks by the Compliance Service.
7. If no regulatory limits are surpassed the trade takes place.
8. The investor no longer has the tokens in their Exchange wallet, and has received the currency/tokens they asked for in exchange.

ADAPTATION TO COMPLIANCE PROTOCOLS

Given how critical regulatory compliance is for the Digital Securities market, it is no surprise that several initiatives are already trying to address this. Harbor's R-Token [1], Polymath's ST20 protocol [2], the ERC-884 proposal [3] for shares compliant with Delaware Corporations Law, and Open Finance's S3 Protocol [4] are examples of the efforts to provide on-chain mechanisms to validate transactions in real time so that only authorized investors can transfer or receive specific Digital Securities.

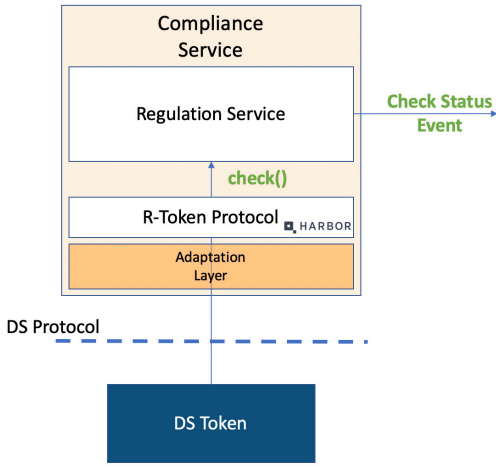
However, there are still problems with the current situation:

- There is still no consensus in the industry about the Digital Securities Protocol. This is obviously a problem for issuers, because they want to launch their tokens in a compliant way but also in a future-proof way. Assets like funds or equities will require their tokens to stay in the market for several years. Issuers cannot afford to choose the wrong protocol.
- Some of the existing approaches also require utility tokens (like POLY or OFN) to integrate in the corresponding network, which generates additional friction in the decision-making process.
- While the existing protocols allow STEs to ensure that no non-compliant transfer of tokens occur, these protocols provide no specific tools for STEs to do these checks beforehand or to add new investors to the authorized investor pool for a given token. This negatively impacts liquidity and the user experience for investors.

Securitize will address this inside the DS Protocol via the implementation of a generic Compliance Service. This generic Compliance Service will provide an adaptation layer to Securitize's DS Tokens for different regulatory compliance protocols.

The Securitize DS Token Smart Contract holds a reference to the Compliance Service, which in turn is able to adapt the validation operations required for any transfer() or transferFrom() operation to the specific format required by existing protocols.

The following diagram shows how Securitize would enable Harbor's R-Token protocol compatibility for a DS Token.

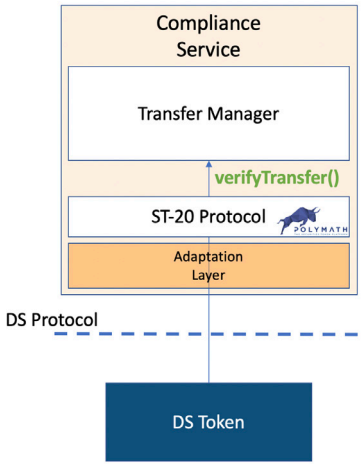


R-TOKEN COMPATIBILITY

Alternatively, the same mechanism could be applied for Polymath's ST-20 compatibility.

As seen above, the Adaptation Layer can transform the invocations coming from the DS Token so that they comply with the check() method defined in the R-Token specification or that they follow the verifyTransfer() method from ST20. It is also possible to support other protocols defined for regulatory compliance.

The DS Protocol enables the upgrading of the underlying modules, which make DS Tokens future-proof from a regulatory perspective, since they can be adapted to new compliance protocols after their issuance and adapt to changes in regulation.



ST-20 COMPATIBILITY

ADDITIONAL INTEGRATIONS WITH EXCHANGES

The on-chain protocol provided by the Trust Service allows Exchanges to be registered as trusted providers of investor identity and accreditation information. This way Exchanges, subject to applicable law qualifiers to deal with reliance and data protection issues, can contribute their own KYC information about their users so that they will become part of a given token investor base and be able to buy and sell that Digital Security in a compliant way. The KYC information will include, among other elements, the relevant country for the investor, the accreditation/qualification status and a cryptographic signature available for audit, to guarantee identification documents kept by the Exchange have not been tampered with after the investor registration.

Yet, this approach may not offer the best customer experience for investors. For example, an investor may register to buy a specific token, only to find they are not allowed to complete the operation because the quota for their investor category has already been covered.

To improve the investor experience, Exchanges can develop DS Apps that – once authorized via the Trust Service – will be able to leverage the services on the DS Protocol to get additional information to improve investor experience.

Additionally, Securitize's platform extends the DS Protocol with the Ready For Exchange (RFE) off-chain API. This API will enable real-time operations to check investor eligibility and to share KYC information from the issuer to the authorized Exchanges, so that they can reduce friction and provide the right user experience in their investor dashboards and additional services.

The RFE API will also be open sourced, so that it can be adopted by other Digital Security issuers. Therefore, the ecosystem for Digital Security issuances and STEs can grow in a sustained way, providing increasingly more liquidity and value to Digital Security investors.

Conclusion

The main focus of Securitize's protocol definition has been to consider the full lifecycle of Digital Securities ownership, beyond issuance, as well as the goals of the different stakeholders: Issuers, Investors, and Exchanges.

The potential around the market of Digital Securities is sufficiently sizeable that any approach needs to consider the multiple actors that will need to be involved. That is why, rather than expecting to control a specific network or marketplace, Securitize wants to help build an ecosystem in which Issuers, DS Issuance Platforms, Investors, DS App developers and Exchanges will be able to unlock the full potential of Digital Securities.

Securitize will provide a reference implementation of this protocol on its own DS Issuance Platform, which will also be supported with the real-world issuance of tokens for its customer base. Securitize is also expecting DS App developers to enrich the ecosystem by providing new capabilities to extend the value that comes with owning Digital Securities for all stakeholders.

References

1. Bob Remeika, Arisa Amano, and David Sacks. *The Regulated TokenTM (R-TokenTM) Standard.* https://harbor.com/rtokenwhitepaper.pdf, 2018.

2. Trevor Koverko, Chris Housser Polymath. *The Securities Token Platform.* https://polymath.network/whitepaper.html, 2018.

3. Dave Sag *Tokenising Shares: Introducing ERC-884.* https://medium.com/coinmonks/tokenising-shares-introducing-erc-884-cc491258e413, 2018.

4. Open Finance Network OpenFinance Network – *White Paper.* https://www.openfinance.io/public/whitepaper.pdf, 2018.

Questions?

info@securitize.io
www.securitize.io

t.me/securitize
twitter.com/securitize

IMPORTANT DISCLAIMER

This “Case Study” is presented as a hypothetical scenario and for illustrative purposes only.

The material has been prepared for informational purposes only, and is not intended to provide legal, tax, investment, accounting or other advice. The recipient should conduct their own inquiries as to the adequacy, accuracy, completeness and reliability of any information, whether such information is contained in this case study or not, relating to issuing a security token via a Digital Security Offering (DSO). The recipient also acknowledges that, to the maximum extent permitted by law, each of Securitize and its related parties or affiliates disclaims all liability to the recipient or to any other person for any expense, cost, loss or damage of any kind including direct, indirect or consequential loss or damage (however caused, including by negligence) incurred by any person arising from or relating to any information included or omitted from this case study, whether by reason of such information being inaccurate or incomplete or for any other reason.