BYZ GEN

ByzGen's DLT backed secure data management and exchange solutions

Keystone White Paper – Version 1.0
July 2020

Terry Leonard
Chief Technology Officer, ByzGen Ltd

CONTENT

ABSTRACT

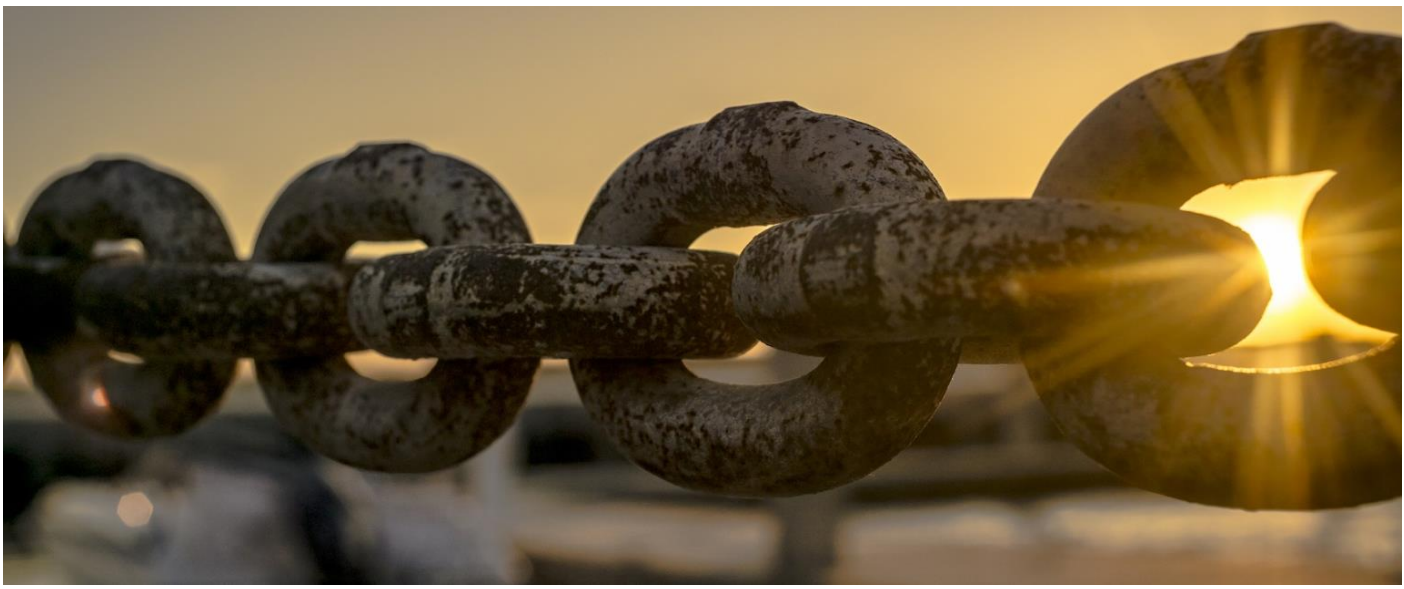
ByzGen have created a variety of bespoke DLT data security products across numerous sectors including defence, manufacturing, and insurance. The products were designed to secure and assure the user's most valuable data types within discreet business processes and data flows.

ByzGen have assimilated the lessons learned from these projects with the perceived industry challenges associated with adopting DLT; and created a fully configurable back end platform that is designed to easily integrate with existing systems and processes. The platform, called Falkor, is now live and Enterprise ready.

To date, the majority of DLT solution providers have focused on the Financial Service's Crypto-currency sector; ByzGen have chosen to focus entirely on the data-driven Enterprise market.

Falkor leverages the benefits of private distributed ledger (replacing crypto currencies and tokens with business-critical data) using APIs and sector-leading security protocols to deliver a unique Trusted Data Exchange and Golden Record capability. The platform enables companies and organisations that have a critical need to exchange, secure and control their most sensitive and commercially valuable data with each other; *inter alia* design IP, trials and test data, AI/ML algorithms and digital assets/value.

This Whitepaper represents 3 years of product development and focuses on the overarching capabilities and functions of the Falkor platform. A series of Technical Briefing Papers have also been written, which complement this Whitepaper. Each Technical Briefing Paper is anchored by a real-world challenge that ByzGen have first-hand experience in helping to solve. As ByzGen deploy our platform to address new challenges, more Technical Briefing Papers will be released.



CAPABILITY OVERVIEW

Falkor is a proprietary platform developed by ByzGen that provides several scalable and interoperable services that allow its clients to deploy, configure and integrate DLT backed data management and exchange solutions to their existing systems, applications, and architecture.

The platform allows those existing systems, applications, and architectures to generically leverage the capability of its services via APIs and container driven dev ops that allows for ease of integration, deployment, and subsequent scaling of the capability enterprise wide and across ecosystems.

The services of the platform combined provide the following principle capabilities:

- use of one API gateway to leverage multiple DLT solutions and remove complexity of these. This allows for flexibility based on the type of data transaction being triggered. Examples of type of transaction would be one that should be validated via proof of consensus versus one that requires two parties to accept an exchange of ownership or value.

- multiple DLT components can be signed to and data and proofs securely transferred between them to maintain integrity and control.

- creates a trusted and auditable data exchange between various other components of existing architecture. These can include siloed application instances, applications running on different cloud projects, accounts, or providers and IOT devices.

- one gateway gives support for multiple projects, datasets and use cases that require the capability of the platform.

- dataset configuration based on type of data being transacted. This is how data from multiple sources is separated and treated in terms of security, storage and control and it can be flexibly configured and scale to dataset types enterprise wide and across ecosystems.

- data encryption that includes a distributed threshold encryption component that provides a layer of additional security and complete immutable audit of when data has been read by any user of a given ecosystem.

- data decryption that includes processing and key generation at the individual read request level and verification against a DLT golden record. This provides a validated access and audit at the exact point at which the data is being requested to be decrypted.

- creation and updating of distributed permissions structures and identities within those structures to maintain the latest validated state of permissions to data across a given ecosystem that will be validated against when a read to data is requested.

- the tracking, enforcement and audit of business processes allowing for verified automation

- creation and maintenance of an ongoing golden record that provides a validated global state that a given network of 3rd parties can operate and be enforced against.

- maintains a validated state of data, value, and ownership within a given network of disparate 3rd parties

- can trigger peer to peer validation for data transactions that requires only the involved parties to sign that the data, value, ownership can be exchanged.

- full immutable audit to the user level of data creation, data versions, permission changes, creation, and update of links between data, reads to data and business process statuses and states. Each of these actions is a transaction on the golden record.

- create, update, and prove that links between data exist without the data itself needing to be decrypted.

- initialisation and deployment of the platform carried out at the individual service level. Each service runs from a containerised image with the full set of images being orchestrated and managed via a central engine.

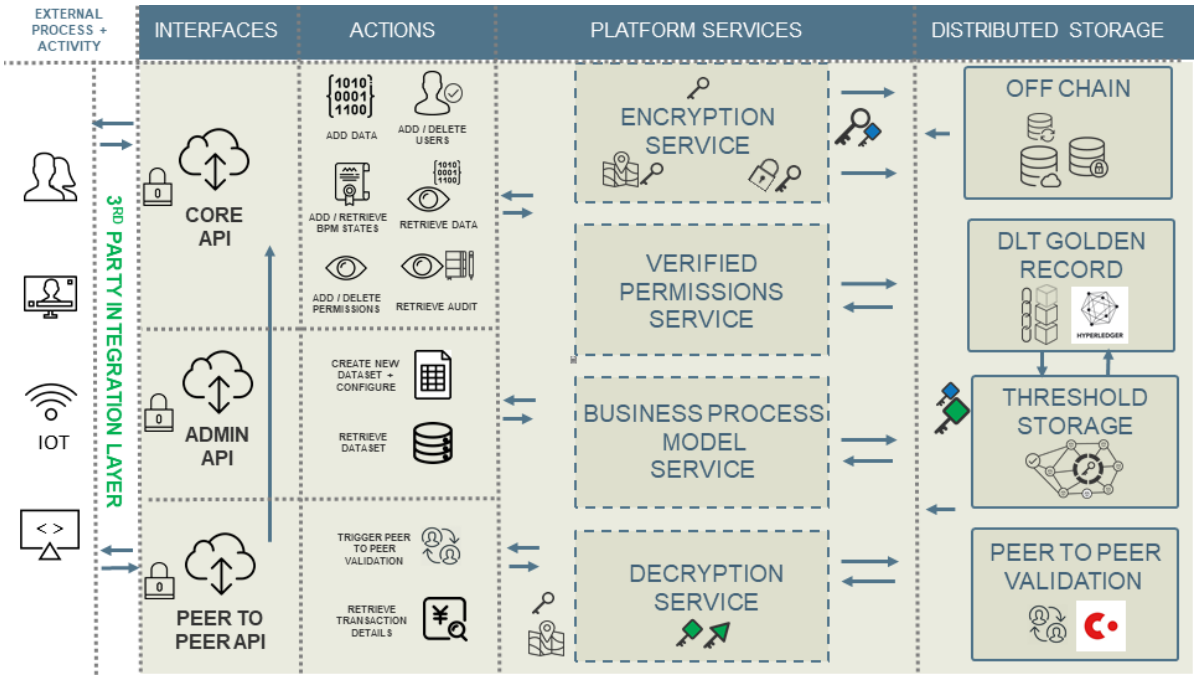
- services can run agnostic of cloud provider, be complex distributed services deployed in a quick time frame and offers flexibility around what services are necessary to be deployed for each use case.

GENERIC PLATFORM PLAY

As we will describe in this document there are several principle and generically leveraged services that when deployed in the appropriate combination will provide the overall capability required for multiple use cases.

This capability can be integrated with to provide enterprise wide coverage from the one integration and gateway.

A generic platform architecture is below:



For any given use case a combination of these services can be deployed to meet requirements, the dev ops mechanisms in place within the platform mean that these can be deployed individually. Each main service has its own docker image and Kubernetes directory to achieve this. The platform uses Kubernetes to orchestrate those images and they are hosted, built and pushed via GitHub actions.

On one side of the scale there could be a use case where metadata needs to be tracked plainly on a DLT in order to provide an audit of a business process without the need to encrypt or have separate write and read permissions for different datasets. This would be a simple deployment that would require only the core API and the DLT golden record from this generic platform architecture.

However, the other side to that scale is there will be full data objects of differing types that require separation, sensitive datasets that require appropriate encryption and control, datasets that require the highest level of security and audit of their decryption, different types of data transaction (peer to peer vs golden record) that can be triggered and multiple disparate parties are required to work with, create and exchange data to create a scalable ecosystem. This is where the other services that work around the core API and DLT golden record need to be deployed. Therefore, having one integration can support multiple use cases enterprise wide.

TYPICAL INTEGRATION AND DEPLOYMENT

The platform is integrated and deployed to solve multiple of the business challenges that exist with data exchange, the activity and transactions produced by data exchange and the business processes that sit around it.

The main business challenges that the platform has been developed to address are described below:

- Mutiple users from disperate parties. These users are not only required to exchange data but also write, read, update, link, and audit the data that makes up those exchanges.

- These users can also be managing business processes that need to be tracked, that produce more data through their workflows or need data as an input to the process. Within these processes there can be multiple statuses, states, and data flows involving several disparate third parties

- Multiple different applications, infrastructures, devices are used by those disperate users but the data and processes to be exchanged and worked on is the same.

- Data can be heavily siloed within party boundaries as a result of legacy systems or internal organisational security procedures

- Datasets to be exchanged and worked on can have differienated requirements about how they should be treated, secured, accessed and tracked. This needs to persist before, during and after the data is exchanged.

- Certain parties can have already decided to start using a certain blockchain solution for their data but the current trend is that this is an internal decision to that party and therefore can create more complexity when trying to transfer data or give access to their blockchain solution of choice generating another silo.

- Links that are created between data that are ever changing which are required to be proven at a given point and as they change.

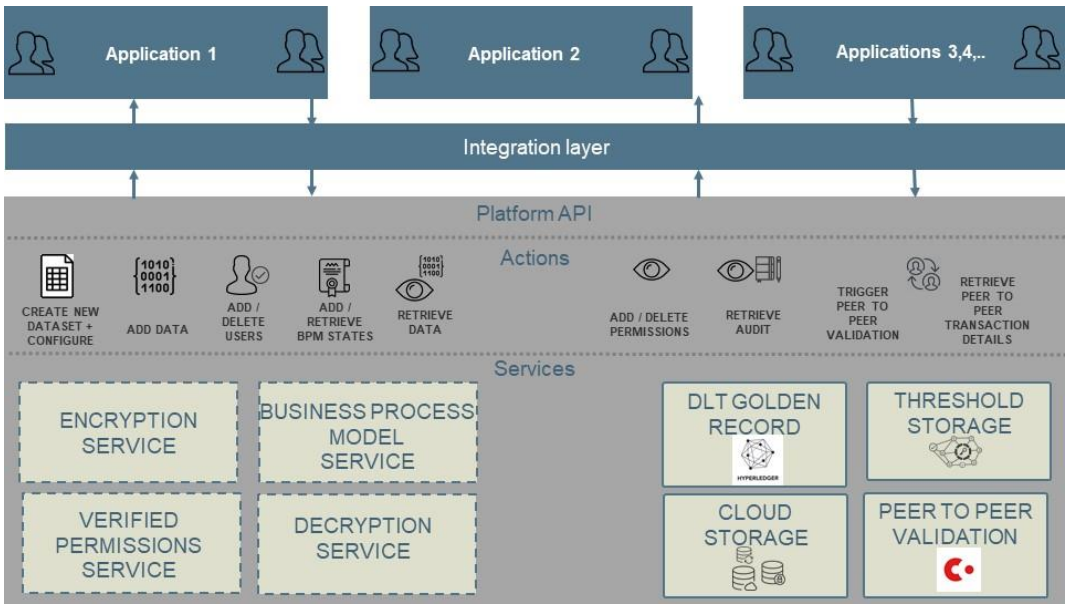
- Continuous transactions between 3rd parties to exchange data, digital assets, and value that change the ownership of those.

- Constant decision making based on current state of data, digital assets, value, and ownership.

- Constant analysis being performed based on current state of data, digital assets, value, and ownership.

- Audit requirement on exactly where data, digital assets, value, and ownership stands at a given point in time and as they change.

Typical Architecture for Integration



Components of Typical Architecture

Application Layer

There will be multiple disparate applications or devices within this layer. Each of these in turn will have several parties and users with in them that are part of a common ecosystem and need to take part in activities such as:

- creation of datasets and data
- control data and versions
- updates to data
- reads of existing data
- management of a business process that has data as an input or output
- exchange data for a given project whilst retaining control
- retrieve audit activity around the data and processes
- enforce proper use of the data produced
- trigger an exchange of ownership to value, a digital asset, or data
- validating an exchange of ownership to value, a digital asset, or data.

These users will need to continue using these applications and their functionality as BAU day to day.

The difference will come when certain datasets are generated that need to be treated with additional control, security, and validation eg: they are going to be exchanged with a third party within an ecosystem.

In this case the application can carry business logic or ML that recognises this and therefore calls out to a different backend to treat and store the data. Alternatively, or additionally the users can manually select the backend option required for the dataset they are creating or working with from the application frontend.

In a simple walkthrough case, a user of one of the applications can create a new dataset which requires the increased level of control, security, and validation. They will set configuration for this dataset in terms of how any data within it should be treated in terms of encryption level, permissions, and end storage or that can be automated based on business logic or ML. Once the dataset exists the user then knows that all data added to it will be treated and validated based on the configuration they have set.

As they then work with that dataset there is other activities that will be transacted as they do:

- Each version of a data object will be tracked incrementally
- links between data can be created and versioned
- versions of data can be validated and retrieved
- links and versions of links can be validated and retrieved without requiring the decryption of the data
- permissions updated and identities added when an exchange is actioned
- retrieval of audit for creates, updates, reads at the version level for data, links and permissions
- trigger a peer to peer transaction to validate that data, an asset or value is going to change ownership
- retrieve latest state of ownership for all data, assets or value

These actions will happen in the background as the user works through their BAU activity on the applications and data is worked on, exchanged and transactions generated.

The assumption is that the applications and user activity already exist in this layer but there is a requirement to bring them together into an ecosystem, be able to bring those together with the platform capability and scale the ecosystem once the baseline is in place.

There is also an option to build out new applications, flows and business logic where required and integrate those to the platform.

Integration Layer

This layer works between the applications being used and the platform API. It can be leveraged by multiple applications concurrently and can take requests from those applications and then translate

those requests to formulate API calls to the platform API. Requests from the applications will be based on the activity being generated by users in their BAU activities at the application level.

Principle Platform Services

These are the generic principle services that are called based on the type of action coming from the platform API. Each service can be deployed separately and can sign to other services within the platform when required.

Platform API

The platform API is the gateway that receives calls from the integration layer and then based on the type of action will then invoke the correct platform services to deal with the request. This is a RESTful service in the JSON data format.

The interface supports and provides endpoints for several actions:

- creation and configuration of datasets – configuration options include encryption type, storage type and permissions **(POST method)**
- creation of data within datasets **(POST method)**
- updates to existing data **(PUT method)**
- retrieve data by version or latest state **(GET method)**
- create links between data **(POST method)**
- update links between data **(PATCH method)**
- addition of permissions for a dataset **(POST method)**
- update of permissions for a dataset **(PUT method)**
- add of id within existing permissions **(PUT method)**
- delete of ID within permissions **(DELETE method)**
- retrieve audit of data **(GET method)**
- retrieve audit of links **(GET method)**
- retrieve audit on permissions **(GET method)**
- trigger a peer to peer transaction **(POST method)**
- retrieve the transaction details of peer to peer transaction **(GET method)**

Each of these calls need to be authenticated via a HMAC generated API key. The API key is required to be passed in the Authorization http request header for any call being made. Such key is linked with a tenant in the platform. Internally, the platform recognises the tenant by the API key provided in the http request header.

The platform can be used as a multi-tenant system and therefore the concept of a tenant is a way of further separation if multiple organisations are connecting to the same instance of the platform and networks and making these requests. A tenantID needs to be passed in all API calls made.

Encryption

When data is added to a dataset that is configured to use encryption then this service will be used at the point at which the data is added. The levels of encryption that are configurable when creating a dataset in the platform are:

No Encryption
If this is configured at the dataset level, then the data added to the dataset will not be encrypted at all before it is stored.

Platform Encryption
If this is configured at the dataset level, then the data will be symmetrically encrypted before it is stored. During this encryption process the data is split into blocks and each block is encrypted separately using Authenticated Encryption (AE / AEAD)

There is then a process to symmetrically encrypt the initial key material that was used to encrypt the data and other metadata linked to the data before that is stored on the DLT golden record and validated as a transaction within a block. The same authenticated encryption method is used within this step.

The 2nd symmetric key that was generated in this case will be further encrypted via a master key and stored separately within off chain storage until it is required in decryption process of the data.

Threshold Encryption

When reads to data within a dataset requires an additional layer of security, validation and audit then threshold encryption can be enabled.

This follows a similar process to platform encryption, however once the 2nd symmetric key is generated this will instead be encrypted via an ElGamal public key that is produced by distributed key generation amongst the threshold encryption nodes. This means that each read of the initial data will require a distributed decryption and re-encryption to take place within those nodes (see threshold encryption section for further details). This in turn generates a transaction that is validated within the DLT golden record for read audit purposes.

This generates flexibility for being able to encrypt data within a specific dataset in the most appropriate way before it is stored based on the sensitivity of data or business logic being employed. It therefore allows for multiple different dataset types to be constantly transacted at scale with the correct security applied to them.

In all these scenarios a checksum is computed over the encrypted data before it is stored. This checksum is stored on the DLT golden record and can be validated at the point at which the data is decrypted to further ensure the integrity of it.

For all symmetric encryption described here the platform uses a block cipher mode of operation that provides high speed of authenticated encryption and protects the integrity of the data until it is fully decrypted.

Decryption

When there is a request to read any data within the platform then this service will be used at the point at which the data is requested.

If a dataset is configured to have full threshold encryption enabled, then an ephemeral key pair will be generated specific to the request that has been generated. Once the IDs and permissions for the unique request are validated within the DLT golden record (see permissions and DLT golden record services) then the threshold encryption service will be signed to and will be passed the ephemeral public key generated for the unique read request. The threshold encryption service will then perform the decryption of the 2nd symmetric and re-encrypt it via the ephemeral public key that it has been passed. Only the threshold encryption service can perform this decryption and re-encryption given it is those nodes that generated the encryption key.

The encrypted 2nd symmetric key can then only be decrypted via the ephemeral private key that has been generated by the platform decryption service. Once the 2nd symmetric key is available it can be used to decrypt the 1st symmetric key and metadata that is being held on the DLT golden record. Once the 1st symmetric key is available then the data itself can be decrypted and made available to the caller of the platform API and through the layers to the applications being used.

Therefore, this process provides a validated access and audit at the exact point at which the data is requested to be decrypted. The level of validation and immutable audit that can be achieved is unique due to the threshold encryption and re-encryption that takes place and that process in itself generating a new transaction in the DLT golden record that can be retrieved.

If platform encryption is enabled for a dataset then the process is similar except that the threshold encryption service is not called. When a read request is validated then a decryption of the 2nd symmetric key takes place via the master key solution.

In all reads of data the checksum that has been stored for a given data version will be computed over the encrypted data that is retrieved for the request to ensure the integrity of the data that has been read.

Similarly reads of data can and will reference the latest global state held by the DLT golden record in terms of data, IDs, permissions, encryption keys and metadata and consider that state before any decryption processes are triggered.

This service is vital for data exchange in terms of validating, down to the version, the data being read at the point at which the request is made and the audit required in circumstances where data is being exchanged and worked on by multiple disparate users.

Verified Permissions Service

Permissions objects in the platform are changeable data structures that are validated and stored on the DLT golden record.

These objects can be created at the dataset level and contain the concepts of owner, writer and reader to data held within the dataset. These structures also hold the identities of users (in the form of IDs) that can interact with data held within the dataset.

Each time a permissions object is created or updated there is a new transaction created that requires validation via the DLT golden record and stored as part of a validated block within it. This provides a constant and latest validated state of the objects that can be enforced for all write and read actions around the dataset and an immutable track of how those objects have changed over time and who has actioned the changes.

During data exchange, users can therefore maintain control over their datasets as they are exchanged and enforce policy around the datasets they have created. This policy can be ever changing with updates always validated and tracked via the DLT golden record. The owners of the dataset can also retrieve the immutable audit of both the latest state of the permission objects and also the historical changes made to it over time so they can make better decisions about how policy should be managed.

The different concepts within the permissions data structures can be used to control and enforce policy in different ways. There is a concept of a “writer” and at the point at which there is an attempt to write something new (new data or version of data) to the dataset then the latest validated state of the permission structure is verified. If the ID of the user making the write request is present in the latest state of the structure then the write is processed, if not then it is rejected. There is a separate concept for a read of data which can contain different identities to write so that policy makers can have flexibility around enforcement of those users that can contribute data to the datasets and those that can read the content of the data within the dataset. The other concept is of an owner, the owners have the rights to set policy and create, update and audit the permission data structure. No matter what concept is being updated there is always a new transaction generated with the newly proposed structure and that new structure needs to be validated via the DLT golden record before it becomes the latest state of the structure that all users and applications within an ecosystem are enforced against.

This capability can integrate with existing IDAM solutions that already exist to manage the users within the applications being used. This kind of integration would mean that access management of individuals within specific organisations continues to be managed by the existing solution but the policy set out by that management can now be enhanced to enforce, validate and audit based on the latest state of data, permissions structures and identities at the very point at which the action to create, update or read data is made. As the roles and access policy of the application IDAM solution changes there is the ability to assert those changes into the platform and have them validated as the latest state that all actions are enforced from.

Business Process Model Service

This service allows for a business process model to be configured, stored, tracked, and executed by a combination of BPM tools and the DLT golden record.

The process model can be mapped and changed separately from the DLT with various touchpoints with the DLT golden record established as the process is moved through statuses and states, and the conditions of those states. This therefore grows the DLT golden record with latest and historical statuses, states and execution of new states. This can then lead to an immutable audit of the statues, state changes of the process and a trusted automation of it.

A process model is configured via the BPM tool and the touchpoints with the DLT golden record can be established. Different states and conditions can be stored on the DLT golden record and as the process is tracked through the touchpoints that generate transactions and the conditions are meet then the different states can be executed. There is then a new active state validated which is what the process is now enforced against for all in the ecosystem. All the executions of a new state will be a transaction on the DLT golden record and therefore the history of state changes and

transactions to know why they were triggered can be included in the immutable audit that is retrieved from the platform.

What was avoided in the development of this service was to code large and complex business processes and store and execute them wholly on the DLT golden record. This kind of solution is often described as smart contracts in the DLT space. The main reasons for this thinking are that business processes can be complex and ever changing and therefore you do not necessarily what to make the whole process immutable and as code on DLT golden record. Therefore, each time the process needs to change and be enforced there is a requirement to transact the full process as code back onto the DLT. This adds complexity and time to that process and carries the risk that older versions of the process could execute on the DLT given they have been made immutable – this is especially if the process is ever changing.

The approach of holding the process model element off the DLT with the touchpoints to the DLT golden record is a way of keeping this flexible and scalable to the complex business processes that work around and are produced by data exchange.

For data exchange these states and conditions can be centred around how and when data is exchanged as various statuses are reached in a business process.

DLT Golden Record (Hyperledger)

The golden record holds in a distributed ledger of all transactions that have been validated via consensus by a threshold of the peers running the ledger. Transactions are grouped into blocks and each block is cryptographically derived from the previous block thus creating an immutable chain of transactions. This service also creates a scalable way that multiple third-party users and applications can all operate and be enforced, tracked against the latest global / golden state which is taken from the latest blocks in the chain. This golden state can change every millisecond and having that ongoing and single source of truth as that happens is increasingly important for data exchange particularly when the number of datasets, data versions and third parties to be exchanged scales.

For data exchange there are multiple actions that will generate a transaction to be validated and stored via the DLT golden record:

- configuration for the dataset when it is created
- based on that configuration of a dataset there will be various transactions generated when new data or a new version is created:

the data itself can be stored on the DLT golden record, either plainly or encrypted depending on the configuration

if encryption is enabled, then the 1st symmetric key used to encrypt the data will be transacted and stored. It will be stored encrypted as per the encryption process.

a unique documentID around data is transacted and stored

if a version of the same documentID is being added then the same ID will be used but an incremented version of that data will be transacted and stored

If off chain storage is enabled then meta data including a pointer for the location of the data will be encrypted, transacted, and stored.

- links between data is a separate concept held on the DLT golden record so that no decryption of data is required to retrieve and prove those links. If a new link is created or current links updated then the new data structure, that will contain the documentIDs of latest links will be transacted and stored.
- If permissions are enabled for a dataset then a changeable data structure will be transacted and stored. The data structure for permissions will include the concepts of owners, writers, and readers whereby ids of users can be added and deleted to each one based on the policy or IDAM changes taking place within the application layer. Once the new data structure is transacted and stored that is what is used to validate reads to data within a given dataset.

These transactions, their validation via consensus and storage in sequential blocks creates a record that can provide:

- Verification at the point of data being requested to be read against the latest

validated state of data, its version, its latest permissions, and identities (owners, readers, writers) before any decryption process is triggered

- Proof of data integrity, data versions, and links between data without requiring decryption of the data itself
- A cryptographically immutable audit of all data, data versions, the actions performed on that data (writes, reads, updates), the permissions changes surrounding the data, the links and updates to links and dataset configuration
- Retrieval at any point of latest validated state of data, versions, links, value, and ownership. Each read creates a transaction on the DLT golden record.

Another function of the DLT golden record is to be involved in the signing and calling to and from other components in the platform when necessary.

For Corda, there can be a secure bridge created whereby the result of peer to peer transactions that are validated via 3rd party corda nodes are created back into the DLT golden record. There is a transaction generated in the DLT golden record for these results and therefore the latest validated state that is enforced and can be retrieved includes the statuses of the peer to peer validations.

For threshold encryption, the DLT golden record is part of the process for the signing and validation of read requests. When a read request is triggered from the platform API then the ephemeral key pair will be generated specific to that request, then a call is made to the DLT golden record with the specific request details which are verified against the latest state held for the data and its permissions. If verified an approveID is generated for this request.

The approveID can then be used to request to DLT golden record for each peer to generate a digital signature certificate and those are collated. A request is then made to the threshold encryption component, the request is verified by checking the digital signature certificate and the threshold encryption component will then perform the decryption and re-encryption of the 2nd symmetric key by the ephemeral public key for this request and this is how that material is passed back to the platform.

NOTE: that the DLT golden record does not directly connect to the crypto nodes of threshold encryption. The platform takes care of the direct requests and signing that needs to take place between those services.

In some cases, the DLT golden record will also have validated and stored encrypted details that go alongside the main data object that has been created. For example, if off-chain storage is configured for a dataset then a pointer to where the data is stored off-chain will make part of those details that can be retrieved from the DLT golden record and decrypted as part of a read request.

For the DLT golden record used in the platform leverages and deploys Hyperledger Fabric 2.1 which allows for a proof of consensus validation of each block of transactions by multiple peers and each peer will carry an identical version of the blockchain of transactions.

This type of validation is most appropriate for generation of the ongoing golden record and validated state held within, therefore if there is a data transaction type that is required to be validated into the DLT golden record then it's Hyperledger 2.1 that will be used.

There are inbuild dev ops mechanisms that allow for the initialisation of new Hyperledger 2.1 networks based on the requirements of the ecosystem. Requirements can include how many private networks are required per tenant or project, number of organisations required to run within each private network, number of peers that run within each organization and the policy of each network in terms of the threshold required for the number peers from each organization that should sign each block of transactions for it to be validated.

This gives flexibility based on the type of business and operational agreements that are in place for a given ecosystem – for example there could be a prime entity in the network that can be given weight in terms of the number of peers running within the specific organization and the threshold of peers that need to sign new transactions.

Peer to peer validation (Corda)

Corda is used when there is a certain type of data transaction that requires peer to peer validation for the outcome to be reached. This type of transaction will typically be triggered when there is an exchange

of ownership that needs to be agreed for a given data object, digital asset or value, or indeed any transactions that should only be signed and validated by the parties involved in the exchange.

In this case the data, digital asset or value can be tracked continually on the DLT golden record in terms of the latest state of the data, updates and versions, permissions, links, and accesses. Then, if a point is reached whereby it can go forward for this type of exchange then a peer to peer transaction can be triggered via the platform API.

When initialising a corda network for a given ecosystem there will be a 3rd party corda node created to map to the number of parties that can be involved in this type of transaction. As part of this initialisation each Corda node will have certificates and keys generated that allow for authentication between nodes of the network.

Each 3rd party node that is deployed can be separated from an infrastructure point of view either via a separate cloud project or be deployed in the infrastructure of the 3rd party itself - this is dependent on the ecosystem requirements and individual situation of the parties involved. There is also a requirement for a platform notary corda node to be part of the network that is created so that transactions can be proposed and signatures from transactions can be collated.

Once the network is initialised then corda smart contracts will be used to ensure the correct flow is in place for all transactions triggered for peer to peer transactions for a given ecosystem. This will be based on the ecosystems business requirements and processes.

In all cases the platform will operate an asynchronous flow whereby the transaction will be proposed, and it will then wait until all parties defined to be involved in the transaction sign and a signature is generated by the 3rd party Corda node that is representing that party.

A typical flow for this scenario will be that a trigger will come from the application layer that a certain subset of data is ready to go through the peer to peer validation processing. The integration layer will then need to confirm to the specific peer to peer API endpoint within the platform API what data from the DLT golden record should be part of the transaction (i.e. the documentID) and what 3rd parties should be involved in the validation of the specific transaction.

The platform peer to peer service will then process that trigger, authenticate to the 3rd party Corda nodes that represent the parties defined to be involved in the transaction and propose a transaction to those nodes. The platform will then wait for signatures from all parties involved in the transaction before any further action takes place – the signatures will be sent to the platform notary node that will collate responses.

At this point there is an option for a separate business process for accepting the transaction to be inserted into the flow. Given the platform always knows the latest confirmed state of all peer to peer transactions for a given ecosystem and the parties involved then an ongoing list of transactions can be made available to a party via the application layer.

Therefore, those transactions can be extracted and made machine readable via an application, from there the party can explicitly accept the transaction via the application and that will be the trigger for the associated 3rd party corda node to sign the transaction and a signature be sent to the platform corda notary node. Once the platform notary node receives all signatures required then the transaction is fully validated on the platform.

Once the transaction is validated then it is stored in the local state db on the 3rd party Corda nodes involved. No other Corda node in the network will be aware that the validation has taken place.

The next step in the process is that the platform peer to peer service will call out to the core API and create a new document within a specific dataset that will confirm to the DLT golden record that the subset of value held in the treaty has exchanged ownership.

This also means that a transaction has been validated on the DLT golden record that maps to the corda transaction being valid which grows the record that can be made available to all users of the ecosystem – including analysts and auditors.

This process creates a secure bridge between a transaction that has been validated peer to peer in Corda and the DLT golden record. That means then

any user on the left side of the diagram that wants to view the latest state of the treaty or subset of value they can confirm that without being involved in the acceptance of the transaction themselves.

The secure bridge is further enhanced by the other platform services that have been described in this document:

Threshold Encryption

The documents used to transact the result of a corda transaction on the DLT golden record can have this full encryption enabled that gives the additional security and auditability of those documents.

Permissions

The datasets where the results of a corda transaction are stored on the DLT golden record can have permissions enabled and a permission structure whereby it is enforced that only the platform user can write documents to it and only certain users in the ecosystem can read the results of those transactions as required.

The 3rd party corda nodes can be deployed within the platform infrastructure with the correct separation, access, and authentication or indeed they can be hosted and deployed within the 3rd party's infrastructure based on ecosystem requirements and current infrastructure in place within the specific 3rd party.

This kind of peer to peer validation can also support other communication or transactions that should be kept between two parties in the ecosystem where the exact details are not required to be known or transacted on the DLT golden record. The main confirmation and results from more secret peer to peer transactions can be flexibly added to the DLT golden record where required to ensure the global state is maintained.

Cloud (Off-Chain) storage

Off-chain storage is an option for when data objects should not or cannot be stored directly to the DLT golden record. To note, even if the data objects final resting place is the off-chain storage then there are always transactions around that data that are validated onto the DLT golden record and therefore the overall capability is still realised i.e. there will be a latest and historical state recorded around versions, permissions, links, encryption keys and other encrypted data including the pointer to where the data is stored.

The current cases whereby the off-chain storage option is configured for datasets in the platform are:

- *Size*

  There is a need not to bloat or decrease the performance of the chain by transacting big data and making it immutable. The technology is not a solution for that.

  So, for performance reasons if the size of the data is over 2KB then the recommendation is for off-chain storage to be configured.

- *Personal*

  Under GDPR legislation you need to be able to delete the personally sensitive data on request from the individual who it pertains to. If you are transacting that full data object on DLT you are making that cryptographically immutable and therefore it cannot be deleted in accordance with the legislation.

  The solution here is to separate out what the blockchain should be transacting and validating around the data and the storage place of the data itself. The personal data object can be encrypted and stored within the configured off-chain storage option. Then, the DLT golden record is used to transact documentID, versions, links, encrypted keys, latest permissions, and other encrypted details pertaining to the data object e.g. the pointer to where the data is stored off-chain.

  Therefore, all transactions that need to be validated and audited still take place on the DLT golden record, but the data object is separately stored.

  This means that if there is a scenario where the data object needs to be deleted under the legislation then it can be but importantly there is a full history of all data transactions around that object that is immutable on the DLT golden record.

IP sensitive

As with the personally sensitive data there is good reason to encrypt and store data off-chain in the commercially sensitive dataset space.

Ultimately though the decision on if off-chain storage is used sits with the ecosystem requirements and the users within it. Via the admin API new datasets can be created at scale with different configuration which includes the storage option.

When creating a new dataset, the user can configure it so that any data object added to the dataset will be stored within the off-chain storage component. The other configuration you can place on the dataset level is encryption type from no encryption, to platform encryption to threshold encryption and you can apply permissions at the dataset level.

Therefore, there is flexibility at scale as the number of datasets grows. The ByzGen recommendation for data that is sensitive (either personally or commercially) is to configure the dataset to have off-chain storage, threshold encryption and permissions enabled. This ensures that the maximum security, control, and audit is placed across that dataset which is especially important when trusted exchange of that data to disparate third parties is required.

Off-chain storage can be any cloud or on-premises storage solution that the platform can be configured to authenticate against when the off-chain storage option is configured for a dataset. A current trend in data exchange is to drive to use cloud-based buckets for this kind of storage given cost efficiency and elasticity.

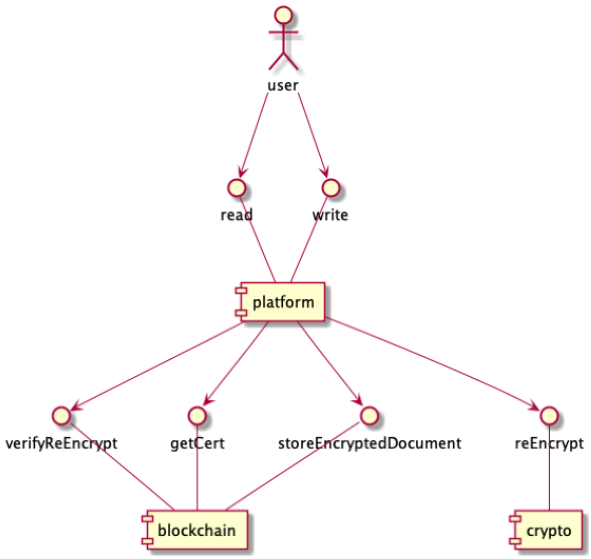
Furthermore, configuring a cloud-based bucket for this option also means that if there is a certain dataset a user wants to move to the cloud they can use the platform to ensure that this data is properly treated as it makes that transfer – i.e it can be encrypted, all actions tracked and validated on the DLT golden record and all it's accesses, versions and links protected and auditable.

Therefore, the platform in this context can facilitate that move of datasets to the cloud.

Threshold encryption (storage)

This service provides an additional security, access validation and audit layer for datasets that it is configured to operate for. This service runs its own set of distributed crypto nodes. These distributed nodes are therefore separate from the DLT golden record and Corda nodes and there is no direct connection between them, it is down the platform to co-ordinate requests to and from these services and ensure that the correct processing and signing is taking place to realise the overall capability and provide that bridge between them.

See diagram below to show this interaction between these services:



As discussed previously it is the DLT golden record (blockchain labelled here) that is responsible for the storing of encrypted data (keys, pointers, the body of the data itself) and providing the validation of if a full decryption can take place based on the latest validated state transacted.

When the access to data is successfully validated then the DLT golden record will issue certificates which are used to perform operations on the threshold encryption services and the crypto nodes that work within it.

During this read request processing, the certificates are passed from the issuer (DLT golden record chaincode) to the threshold encryption service and the platform is responsible for the requests and the processing of this workflow.

Writing data with threshold encryption

The threshold encryption service will be called as part of a write request to the platform if the user is attempting to write data to a dataset with threshold encryption enabled.

This service will be called once the data object has been symmetrically encrypted, a checksum computed over the encrypted data object, a second symmetric key is generated to encrypt the first symmetric key and the encrypted data object and other encrypted material is stored based on the configuration of the dataset. Note: the 1st symmetric key will always be stored encrypted on the DLT Golden record.

It is the 2nd symmetric key material in this process that is then encrypted further via the threshold encryption service. The service will generate a public ElGamal key (with Curve25519) for threshold encryption, this key is generated via a Distributed Key Generation (DKG) protocol by the crypto nodes of the threshold encryption service. Once encrypted via this ElGamal key and stored the 2nd symmetric key cannot be decrypted unless the crypto nodes of threshold encryption reach a threshold to generate the private element that can be used to decrypt it.

Therefore, to ever decrypt the data a threshold needs to be reached by multiple distributed nodes and so no one entity or node can be responsible for decryption of the data and this is the additional level of security that is provided by this service.

This is further enhanced by the requirement that any read request being made to the threshold encryption service needs to pass certificates issued by the DLT golden record peer nodes belonging to multiple organisations so it is not possible to get access to data without getting control of the majority of organisations participating in that network as well as a threshold of the crypto nodes running the threshold encryption service.

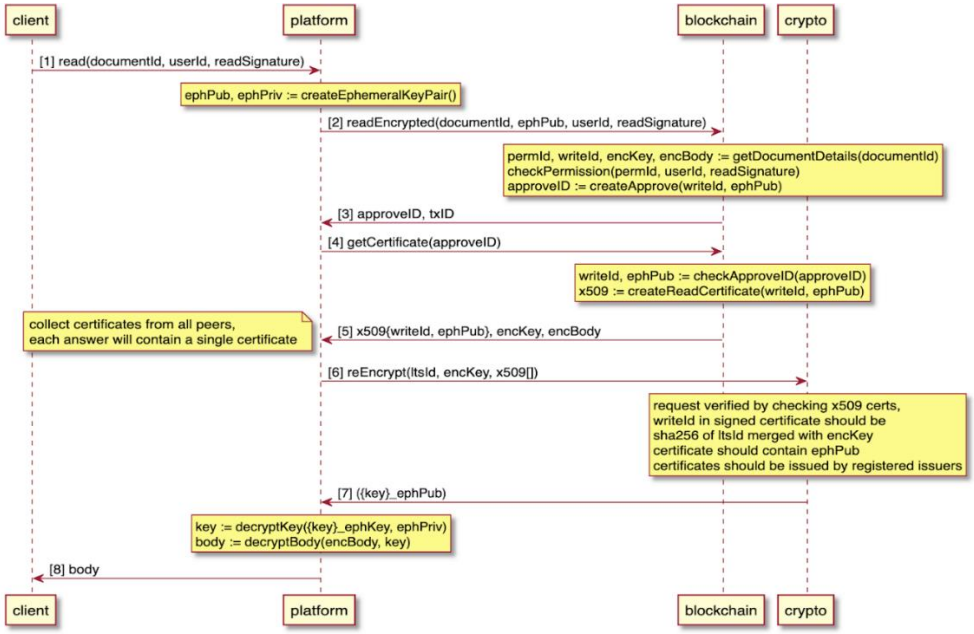
Reading data with threshold encryption

The threshold encryption service will then be called during a read request to any data that is added to a dataset which has this encryption enabled and therefore has a 2nd symmetric key that is encrypted via the ElGamal key generated by the crypto nodes.

Before this service is called during the read process there will have been some steps already completed, these include; the generation of ephemeral key pair unique to the read request, a validation against the latest identities and permissions held on the DLT golden record and the request for certificates to the peer nodes of the DLT golden record in multiple organisations.

The threshold encryption service is then called, and the crypto nodes can decrypt the secure data (in this case the 2nd symmetric key) and at the same time encrypt it by the ephemeral public key delivered in the request made to this service by the platform.

This process of decrypting and re-encrypting data is performed in a way that guarantees that any single crypto node cannot reconstruct the original data. To get this information the private ephemeral key for the given read request is required. This private part of the key is never sent to crypto nodes.



The re-encryption flow that involves the platform, blockchain (DLT golden record) and the crypto nodes is displayed in the figure above. It is started by a read request from the user side via the platform API. Such read request contains the documentID for the data being requested, a version of the data (if applicable), the ID of user who is trying to read the data, and a signature which can prove the origin of such request.

When the read attempt is received by the platform a new ephemeral key pair is generated that is specific to this individual read request. This key pair will be used only for this unique single request and then it will be thrown away by the platform.

The ephemeral key pair generated is an asymmetric elliptic curve crypto key in ED25519 form. The private key of this pair will stay in platform memory and it will never be stored, and it also never be transferred outside of the platform to any of the other services. At this point the platform is sending a readEncrypted request to the DLT golden record (blockchain). Such request contains documentId, dataset, version, tenantID, ephemeral public key, userID and signature.

At this point the DLT golden record validation processing is started, and each peer node is receiving the same input (same readEncrypted request) and each of them by itself is verifying the latest validated read permissions to the data that is being requested by the user.

When the read attempt is successfully verified each then each DLT golden record (blockchain) peer node is creating an approval for this request. At this point each peer node is creating an internal document which is an approval of the reencrypt operation that will be performed by the crypto nodes of the threshold encryption service. All peer nodes (or a threshold of them based on policy) are creating this approval in the same way and all of them propose it as transaction for a next block in the blockchain.

The response from the peers contains an approveID and additionally the encrypted document body and symmetric key encrypted that can be used later to perform the full decryption of the data. The platform then sends the approve ID to all peer nodes. The chaincode on the DLT golden record (blockchain) peer node is then verifying the approve ID and it creates a certificate which will be used by the crypto servers of threshold encryption. This is a regular x509 certificate. The certificate issued by this chaincode contains a writeID and the ephemeral public key previously generated by the platform - both values are stored as x509 extensions. Each DLT golden record (blockchain) peer node holds its own private key (and self-signed root CA certificate). Each peer is playing the role of certificate authority and each of them can issue a x509 certificate. All this logic is handled in the platform created chaincode that is running on the DLT golden record. In the result the platform will collect several certificates issued by the DLT golden record peers, each certificate will contain the same writeID and ephemeral public key and each certificate will be issued by a different peer. These certificates are valid for only a short period.

The collected certificates, the additionally encrypted key and the ID of threshold encryption service installation is sent to that installation of the service. The threshold encryption service is then verifying the request and propagates this request to the crypto nodes. The crypto nodes are then verifying the certificates (i.e. verification of the certificate issuer, writeID, validity time). Depending on the threshold configuration, a minimum number of certificates are required for the crypto nodes to accept the re-encryption request. When this verification is successful the crypto nodes are converting the encrypted 2nd symmetric key to the key encrypted by the ephemeral public key.

As described the platform is the only holder of the private element of the ephemeral key pair so only the platform can decrypt the material received from the threshold encryption service. Once the symmetric key is decrypted it can be used to decrypt the encrypted material for the data that was held on the DLT golden record. Each re-encryption will generate its own transaction within the DLT golden record that can then be retrieved via an audit request.

In summary, this service and the processing and transactions it generates is used in data exchange to provide an additional layer of security and audit for access to data. Therefore, if a user has a particularly sensitive (personal, commercial etc) dataset then the recommendation is that this dataset should be configured to use the additional threshold encryption service and processing. This is particularly when the data is going to be exchanged and used by disparate rate 3rd parties as this transfer will require the highest level of security, validation and audit provided.

HOW BYZGEN WORK & FIT INTO THE DISTRIBUTED LEDGER SECTOR

There are some key values and approaches that ByzGen hold to when developing solutions in the DLT and primarily blockchain space that has shaped our platform capability and associated client engagements:

- Investigate multiple solutions in terms of DLT and blockchain and have no motivations to back just one of them and be too dependent and inflexible as a result. The route taken has been to gain knowledge of what each potential solution could bring to the platform and integrate those as part of the wider capability if they can provide a real differential for use cases and clients of the platform. These solutions are integrated in a way that they should work alongside the other components of the platform and be deployable via the central dev ops mechanisms of it. This has led to us working in the academic research space and leveraged our threshold encryption service from that and we have also integrated the ever-maturing open source solutions of Hyperledger and Corda.

- Each of these bring different and required capability to the platform. The threshold encryption component uses the concept of a distributed network for validation but looks at it through the lens of multiple nodes being involved in the encryption, decryption and re-encryption of sensitive data and therefore provides a unique level of security, validation and audit around sensitive datasets. If this level of capability is available, can be leveraged in a non-complex way and is performance ready then there would need to be a question of why it would not be used by a client. The Hyperledger component gives us the proof of consensus validation and block creation that provides the benefits of the DLT golden record described in this document. The Corda component gives the option for triggering a differing type of data transaction whereby an exchange or change of ownership is required to be validated by only the parties involved in the transaction.

These individual components are brought together via in-built platform development and are under the control of the platform rather than any third parties e.g in the scenario where corda needs to transact results in the DLT golden record, the golden record needs to provide proof that a read request is valid before generating approval certs for use in the requests to the threshold encryption component.

- A primary approach is to have the platform be a non-complex gateway to multiple DLT solutions, the differing capability they bring, the bringing together of them and the running and deployment of them. This means a client does not need to be burdened by the complexity and undertake development for the integration to each solution. They can instead leverage the platform API in a generic way and serve multiple projects and use cases they have enterprise wide. This is in particular relevance when you see the different data transaction types that are required to be supported and the need to leverage different DLT solutions to provide the golden record versus the pure peer to peer validation for certain transactions.

- Build non DLT components that can work agnostic to DLT solution that are proven to be required (outside of generic blockchain) during the build of real use cases and solving of real business challenges. Examples of these are to have encryption and decryption processes before and after data is stored, dataset configuration that is scalable and flexible, enhanced security and audit for reads to the data, being able to create, update and store links between data and other properties without needing to decrypt the data itself.

We believe that generic blockchain solutions only are not a sole solution to many use cases in themselves and these other non-DLT components need to operate around them. Also, not one generic blockchain solution provides all the capability and therefore if multiple of them are to be leveraged

then these other non-DLT components will need to deal with requests between them and the secure bridge of those requests and data that passes between them.

Part of this is always to question what processes and data should and should not be transacted on DLT. Once you have transacted data onto DLT you have made it cryptographically immutable and this can be of great benefit when you want to validated against a latest state of data, permissions, track versions / links and produce an audit for those. However, in the scenario where there is personally sensitive data (under GDPR legislation) involved then transacting that data directly to a generic blockchain solution makes that data immutable and therefore it is unable to be deleted which goes against the legislation. Also, in the case of a business process model there is care required when attempting to code the whole process and make it immutable via smart contracts on a blockchain, this can cause problems when that process changes and there is a potential that an old version of the process could execute.

With these non-DLT components the platform can also be used to solve the challenge of different organisations who have chosen and implemented certain blockchain solutions and are therefore potentially creating more data silo challenges when trying to make data exchanges with 3rd party organisations who have made a decision to implement another blockchain solution. As proven with the processing between corda and the DLT golden record through the platform then these components can be used to bridge the gap between use of different blockchain solutions.

- We work for our clients. This is multi-faceted in approach and means that platform development is based on proving against real use cases and business, there is no bias on blockchain solution, focus on taking complexity away from the client, to prove and ensure new functionality can be deployed agnostic of cloud infrastructure and produce generic platform functionality.

- Be an enterprise wide capability and therefore always consider scale, flexibility, performance, and deployment when developing new capability.

- Continually research and innovate around new capability that can be tested, proven and integrated with the platform and make them readily available

PERFORMANCE (SCENARIO-BASED / TRADE-OFFS)

Following the move to the latest Hyperledger version (2.1), performance testing is underway. Results will be re-published in V1.1 of this paper (expected release before end Q3 2020).