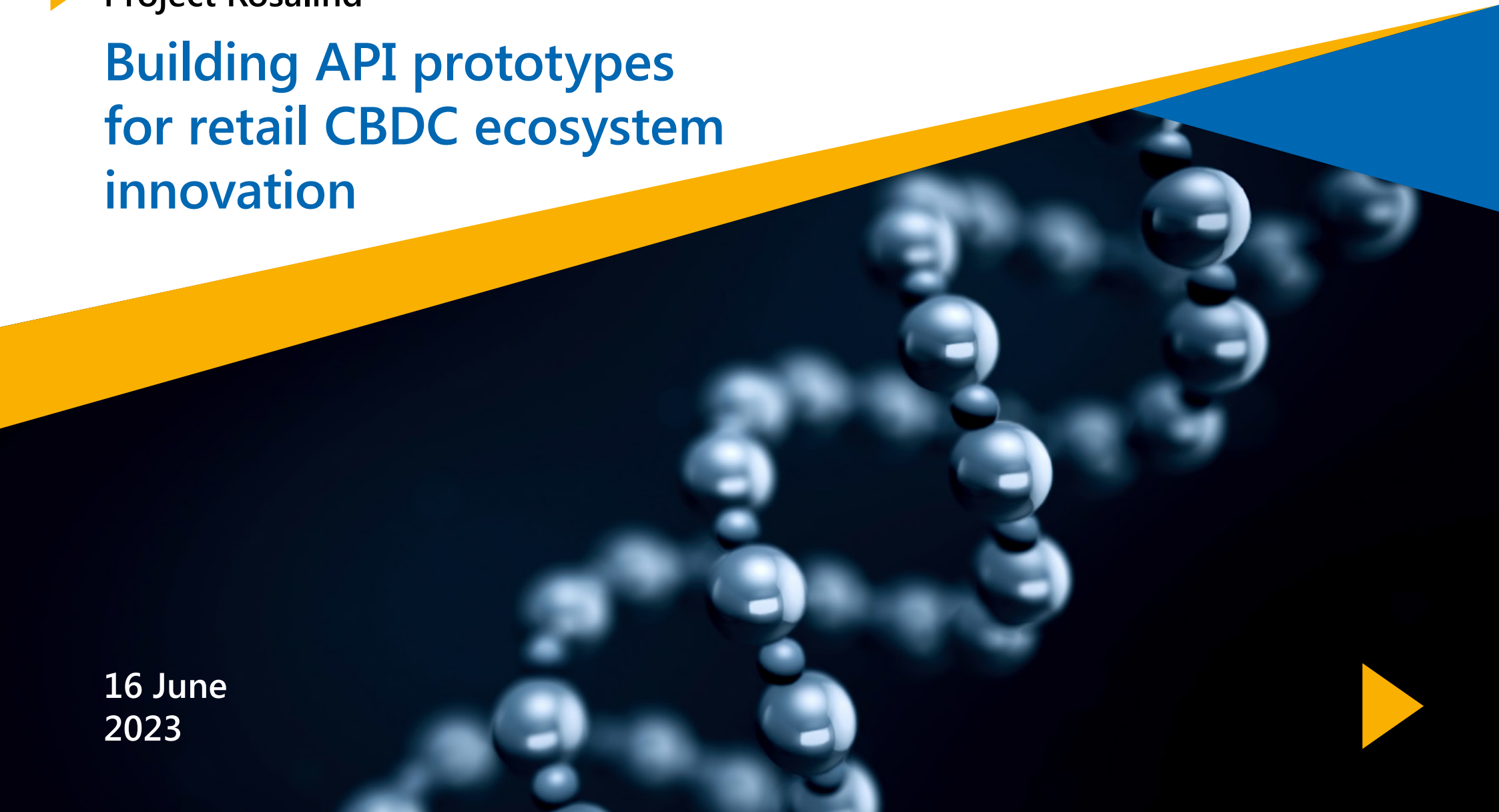
BIS Innovation Hub

Bank of England

Project Rosalind

Building API prototypes for retail CBDC ecosystem innovation



16 June 2023

Contents

Glossary of terms

AML	anti-money laundering
ATM	automated teller machine
API	application programming interface
BIS	Bank for International Settlements
BIS CPMI	BIS Committee on Payments and Market Infrastructures
CBDC	central bank digital currency
CTF	counter-terrorism financing
DID	decentralised identifier
ESIP	ecosystem service interface provider
FAPI	financial-grade API
FSB	Financial Stability Board
HTLC	hash timelock contract
LEI	legal entity identifier
JSON	JavaScript Object Notation
JWS	JSON Web Signature
KYC	know your customer
NFC	near-field communication
NFT	non-fungible token
POS	point of sale
PIP	payment interface provider
PII	personal identifiable information
REST	representational state transfer
RTP	request to pay
ARTP	authenticated request to pay
TLS	transport layer security
UTXO	unspent transaction output
VC	verifiable credentials

Executive summary

Project Rosalind is an experiment exploring application programming interfaces (APIs) for retail central bank digital currency (CBDC).

The project is based on a two-tier model representing a public-private partnership in which the central bank issues CBDC and provides the ledger infrastructure, and the private sector offers user-facing services including digital wallets.

At the centre of this architecture is an API layer, which connects public and private infrastructures. The API layer offers a set of standardised functionalities to enable different systems to interoperate. The project explored how central banks could address the need for a universal and extensible API layer for retail CBDC payments. Collaborating with the private sector, the project also explored what the building blocks of a CBDC ecosystem would be and how the APIs could support innovative use cases.

The project involved the development of a prototype API layer, with 33 API endpoints in six functional categories.[1] The design and functionalities of the APIs were tested and validated through more than 30 use cases identified and explored by public and private sector collaborators. A global showcasing event and a TechSprint were important parts of the project for deepening understanding, sharing progress, and engaging with the ecosystem.

The key findings were:

- A well-designed API layer could facilitate retail payments in CBDC.
- A set of simple and standardised API functionalities could support a diverse range of use cases. It also has the potential to support innovation in products and services based on CBDC that could help meet the future needs of users in a more digital economy.
- The API layer could work with different central bank ledger designs to facilitate payments.
- The design of the API layer must be consistent with, and implement the requirements of, the wider privacy model for a CBDC. This was fundamental to the design and build of the Rosalind APIs.
- APIs can support offline payments in CBDC, but there are a range of challenges involved in delivering offline functionality.

The project also highlighted several areas for further exploration, related to both technology and policy considerations.

- Important considerations emerged around how the APIs might allow the ecosystem to share user and payments data in a privacy-preserving way, subject to user permission.
- When designing APIs, there was a trade-off between extensibility and consistency. The project aimed to keep APIs as simple and standardised as possible whilst enabling service providers greater flexibility to build bespoke features for their specific use cases. This approach could help to maximise extensibility and support innovation, but it might not deliver a consistent experience for individuals and businesses.
- Furthermore, as the project tested the technological feasibility of an API layer capable of connecting systems, coordinating activities and facilitating payments, it identified the need for further work to define the operational roles and responsibilities of all participants in the ecosystem.

1 Introduction

1.1 The global landscape

Central banks are accelerating their explorations of retail CBDC.[2] Many of them are trying to understand how CBDC could support a more digitalised economy and improve the availability and utility of retail central bank money in a digital age, while sustaining confidence and trust in the monetary and financial system.

These central bank explorations are supported by numerous initiatives and projects, from both the public and private sectors, covering a wide range of topics, such as resilience, cyber security, offline payments, privacy protection and ledger design. Authorities are also considering the implications for operational, legal, and regulatory frameworks for CBDC.

Building on the knowledge gained from wholesale CBDC projects, the Bank for International Settlements (BIS) Innovation Hub is now spearheading several technological experiments related to retail CBDC. Experiments cover CBDC architectural models, cyber security, resilience, offline payments, privacy, and cross-border payments.[3]

1.2 The two-tier CBDC model and the use of APIs

Two options for retail CBDC models have emerged: a single-tier system operated by the central bank, and a two-tier model where the central bank provides a core infrastructure, with user-facing services provided by private-sector service providers.[4] One of the key features of the two-tier model is the need for a clearly defined and well-functioning public-private partnership.

There are a range of possible designs for a two-tier retail CBDC system, with various types of interaction between private and public sector participants, and different boundaries between public and private sector infrastructure.

Project Rosalind is based on a version of this two-tier model. It would enable individuals and businesses to manage their CBDC balances held at the central bank and recorded on a central bank ledger, and to make payments through their private sector service providers. Those providers, including payment interface providers (PIPs) and ecosystem service interface providers (ESIPs), might be banks, financial institutions, or non-financial institutions provided they have the appropriate regulatory status and permission to offer such services. See Section 3.9 for details on the service providers explored in this project. All transactions would be settled on the central bank ledger in real-time, on a one-to-one basis, without bundling or netting with other transactions, and with finality.

This model would require an API layer which would help to pass instructions from service providers to the central bank ledger, and orchestrate activities to initiate, verify, approve, and finalise payments. An API is a set of defined rules and protocols that are used to allow applications and systems to communicate with each other. APIs can act as interfaces between different systems in order to process requests to perform specific tasks or access specific data. They can improve functionality and interoperability between payments systems, while abstracting away the inner workings of individual systems. In the past decade, the use of APIs has gained prominence for supporting innovation and contributing to the development of new products and services. The United Kingdom's Open Banking programme is one such example. APIs could also play an important role in cross-border payments.

In the G20 roadmap, the need to harmonise API protocols for data exchange was identified as one of the building blocks for enhancing cross-border payments.[5] The BIS CPMI's report to the G20 in 2022 also discussed the benefits of adopting APIs for payment systems and highlighted the need for API standardisation.[6]

2 Project Rosalind

This chapter describes the purpose and objectives of Project Rosalind in Section 2.1 and the delivery approach in Section 2.2.

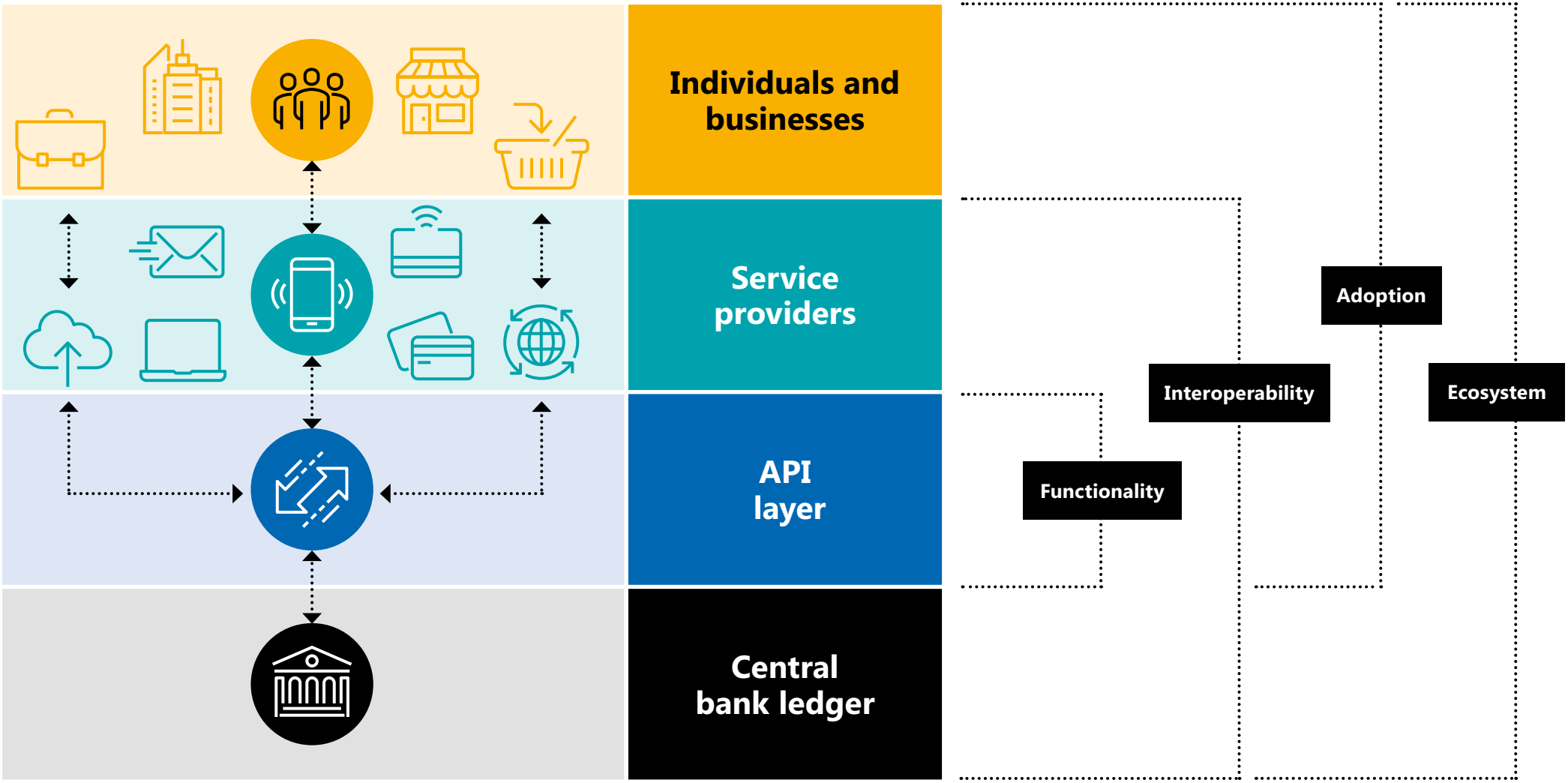
2.1 Project purpose and objectives

The purpose of Project Rosalind was to explore how APIs could be used to support the functionality, adoption and innovation of CBDC systems by developing prototypes for an API layer in a two-tier retail CBDC model.

The project had the following objectives:

- **functionality –** to explore how APIs might best enable a central bank ledger to interact with private sector service providers, including the different options for facilitating safe and secure retail CBDC transactions;
- **interoperability –** to explore how interoperability between different systems and applications could be achieved, including seeking insights into the different design choices, risks, opportunities, and trade-offs involved in delivering interoperability;
- **adoption –** to explore the API functionalities required to enable the development of a diverse and innovative set of CBDC use cases; and
- **ecosystem –** to gain insights into how public and private sector participants could work together to innovate, support digital inclusion, provide diverse payment options, and deliver good consumer outcomes.

Chart 1 below illustrates how these objectives link to the two-tier CBDC model.



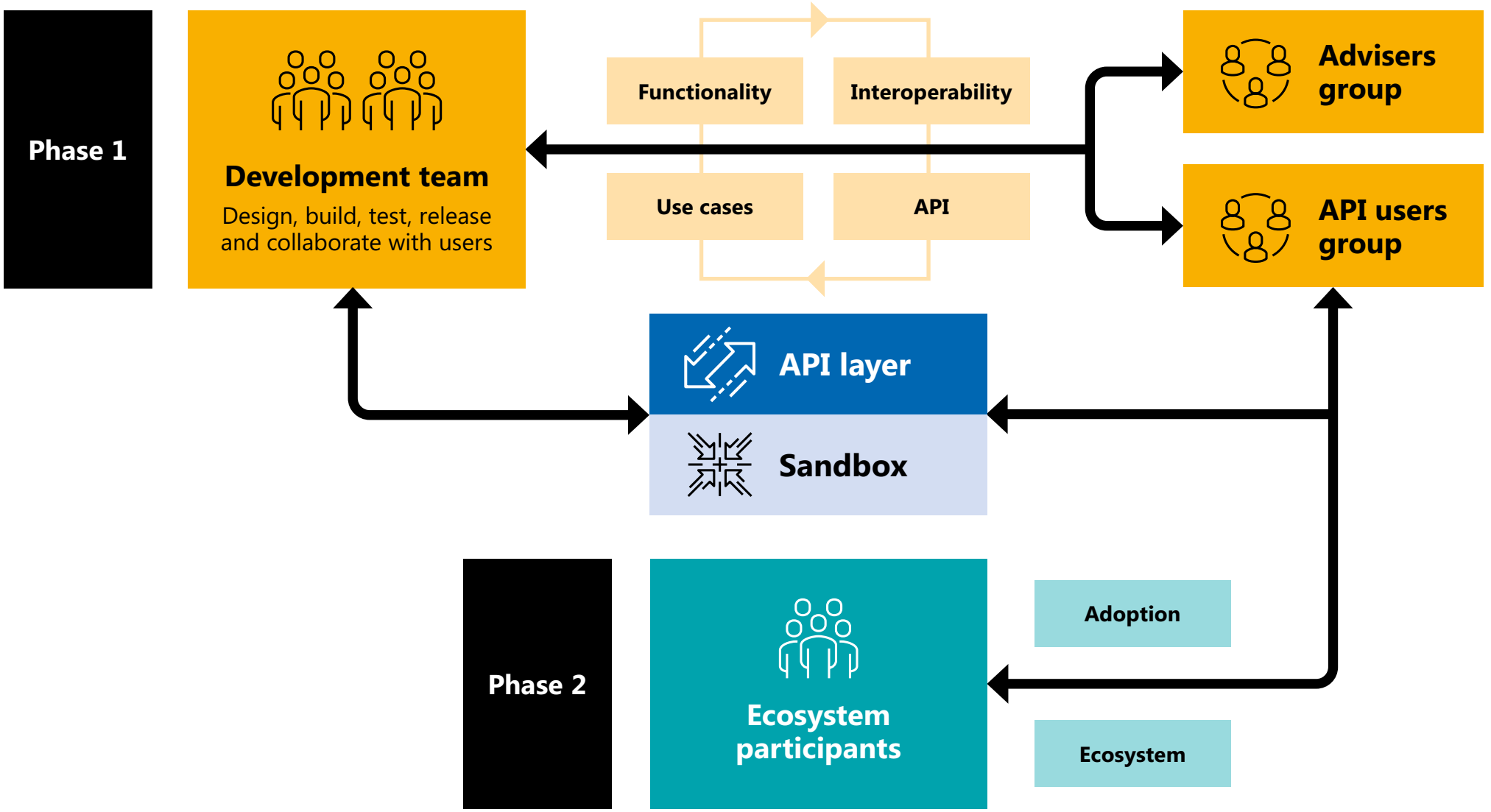
2.2 The collaborative approach

The project was delivered through public and private sector collaboration. Guided by design thinking principles and methodologies, the project created multiple channels for ecosystem engagement, as well as spaces for testing new ideas and options. The project was split into two phases to allow for different levels of ecosystem participation.

Phase 1 focused on designing and developing the API prototype and testing its functionality with developers (API users group) and industry experts (Advisers group). This phase culminated in a showcasing event. Phase 2 explored more use cases by engaging more widely with the ecosystem. In March 2023, the Rosalind TechSprint was launched. Twenty-three teams participated and demonstrated a diverse range of CBDC use cases. In April 2023, selected teams presented their solutions to a group of central banks at a demo day event. The list of participants for both phases can be found in Appendix 3.

This collaborative approach helped ensure the API layer prototype was designed and built with potential user needs incorporated. During the project, the Rosalind APIs enabled collaborators to prototype solutions for more than 30 potential use cases. Details on some of these use cases can be found in Chapter 4 and in Appendix 2.

Chart 2 below illustrates how the two phases were structured and the channels for collaboration.



3 Technical design and build

This chapter presents an in-depth view of the various technical aspects of the project. Sections 3.1 and 3.2 describe the project architecture, key assumptions and design decisions for the two-tier model. Section 3.3 discusses API design principles. Section 3.4 describes all the API functionalities developed. Sections 3.5 to 3.9 cover other topics, including account- and token-based central bank ledgers, the privacy model, security, standards, and the role of, and interactions with, service providers.

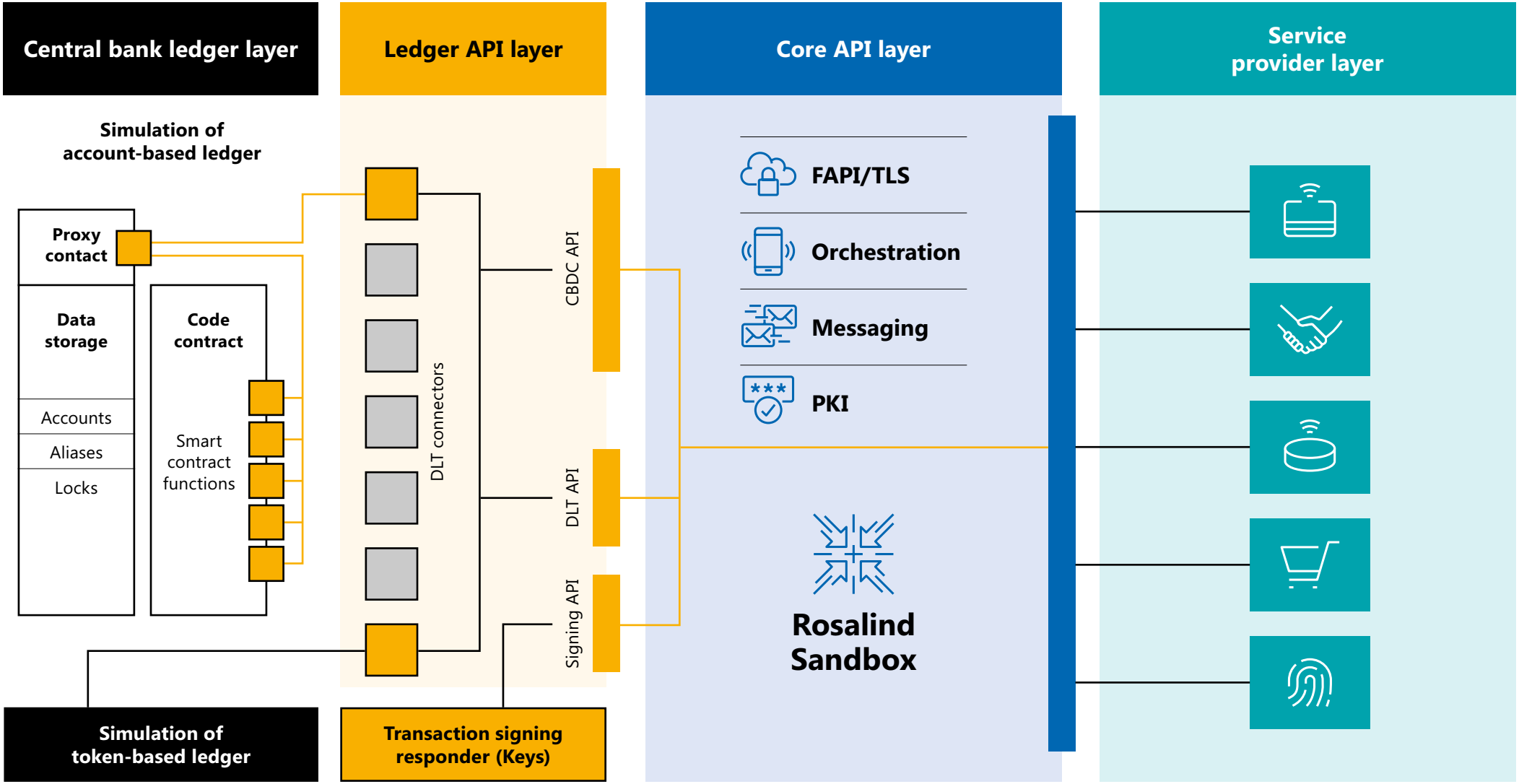
3.1 Project architecture for the two-tier model

The Rosalind architecture consisted of four layers: a central bank ledger, ledger API, core API and service providers. The focus of the project was the core API layer because this layer connected public and private sector infrastructures. It played a critical role in supporting functionalities, enabling interoperability, and facilitating use case discovery.

As a result, the exploration of different technology choices for a central bank ledger and the ledger API was limited. The service providers layer was supported through the work of API users and TechSprint participants. These layers were implemented so that interoperability could be tested, and the output of this project could be examined on an end-to-end basis. Details on the four layers are set out below.

- **Central bank ledger layer:** This layer simulated the central bank ledger. Both account- and token-based ledgers were simulated to test whether the core API layer could simplify and abstract away differences in ledger structures. The account-based ledger used Ethereum-based Hyperledger Besu and a proxy contract model for smart contract upgradability. It included data storage and a series of smart contracts to implement the functionality. The token-based ledger used the Hyperledger Fabric blockchain, and a smart contract was implemented to mimic an unspent transaction output (UTXO) model. See Section 3.5 for key findings on how the API could support both account- and token-based central bank ledgers.
- **Ledger API layer:** This layer translated smart contracts into API calls and transformed API requests into a format understood by, and actionable for, the central bank ledger. The project used Overledger technology to accelerate the development of multiple central bank ledger simulations for comparison. It also enabled the project team to explore how the API could work with different types of central bank ledger.
- **Core API layer:** This layer was the focus of the project, as it orchestrated messages and activities through the API endpoints developed during the project. It supported end-to-end encryption between service providers so that no personal identifiable information (PII) was passed to the central bank. All APIs were compliant with financial-grade API (FAPI) standards and with transport layer security (TLS) authentication for secure communication. This layer also included the Rosalind Sandbox. The Sandbox was a developer portal and central repository of technical documentation to support private sector innovation. During the project, the Sandbox provided the environment to support API users and TechSprint participants.
- **Service providers layer:** Interacting with the core API layer, this layer included front-end CBDC use cases developed and tested by API users and TechSprint participants.

Chart 3 below presents the project architecture for the two-tier model.



3.2 Assumptions and design decisions

In order to prototype APIs for the two-tier model, the following assumptions and design decisions were made.[7]

- CBDCs are direct claims on the issuing central bank. Service providers would not issue their own liabilities, nor would they undertake any custody, warehousing, or issuance of "synthetic" CBDCs.
- CBDCs would be unremunerated and could be converted one-to-one into commercial bank money.
- There would be no limits to the quantity of CBDCs held in each user account or permitted in each payment transaction, nor on the aggregate level of issuance in the entire CBDC system.
- Individuals' and businesses' wallets would be provided by PIPs. Those PIPs would be responsible for complying with the relevant anti-money laundering (AML) and counter-terrorism financing (CTF) requirements. PIPs would also be required to keep records of their users' transaction history, as this would not be provided by the Rosalind APIs.
- All transactions would be settled on the central bank ledger in real-time, on a one-to-one basis and, without bundling or netting with other transactions.
- PII and transaction information would not be visible to the API or the central bank ledger. This information would be stored at the PIP level and encrypted when passing through the API layer. Section 3.6 provides more details on this privacy model.

3.3 API design principles and assessment

A number of API design principles were explored during the project and an assessment of the approach taken by the project against these principles is detailed below.

Table 1: API design principles and assessment

Principle	Approach in Rosalind
The API should use industry standards and support secure and safe payment transactions.	▸ Industry-standard REST APIs were used for the API endpoints. In order to describe the Rosalind APIs in a consistent manner and to make it easy for users to test and apply relevant tools, the project used OpenAPI specifications to define the structure and syntax, and JavaScript Object Notation (JSON) for data format. The OpenAPI specifications are a widely adopted language-agnostic way to describe APIs. The JSON data format is known for simplicity, versatility, and user-friendliness. For API names, nouns and plurals were used. To ensure secure and safe payment transactions, the project implemented FAPI authentication, with attributes in ISO20022 format. Section 3.7 below provides further information on authorisation and authentication.
The API should enable interoperability between different systems and technologies.	▸ The project experimented with both account- and token-based ledgers and concluded that the API had the potential to support central bank ledgers with different underlying technologies (see Section 3.5). The API also supported more than 30 use cases explored in Phase 1 and Phase 2, demonstrating the potential to interoperate with different private sector systems and applications (see Chapter 4).

Table 1: API design principles and assessment (continued)

Principle	Approach in Rosalind
The API should be standardised.	▶ The project aimed to develop a standardised set of APIs that could deliver the core functionalities required to support a variety of different use cases. The definition of core and non-core API functionality was explored. For example, the project assessed whether opening a sub-account – which could be linked to a main account – should be a non-core functionality. In the end, it was implemented in the Rosalind APIs because this type of account's relationships could have a broad use (eg separate business accounts, parent and child wallets) and a standardised API could help to ensure consistency in its application across the ecosystem. ▶ The project also explored the question of how to build "right-sized" API services, eg when multiple functionalities should be combined into a single API call. A form of A/B testing was used for this. Two APIs were provided to users and their usage was assessed. The first API returned the total amount of CBDC held in a user account, and the second API returned both the total and unlocked amount of CBDC held in a user account. During the project, the second API was used twice as many times as the first API. ▶ Standardisation was also applied to how user and transaction data are captured, transmitted, and used. Given the project scope and the privacy model implemented (see Section 3.6 below), the project tested how cut-down versions of the ISO20022 messages could be used by service providers to exchange data.
The API should be extensible.	▶ The project explored extensibility. The Rosalind APIs were designed in a modular way, where a set of simple, basic and standardised APIs could be combined in a variety of ways to support more complex and advanced use cases. The functionality of each API was narrowly defined and independent from other APIs. This approach makes it possible to add new capabilities and functionalities with minimal impact to existing APIs. This modular design demonstrated its potential to support multiple innovative use cases during the Rosalind TechSprint. ▶ To maximise extensibility, the project allowed service providers to add bespoke functionalities over and above the standardised set of APIs provided by the core API layer. Whilst this approach helped to broaden the scope of innovative use cases that the API was able to support, it might potentially compromise consistency in user experience, creating hurdles for adoption. More consistency would require a higher level of standardisation in API functionalities, and less flexibility for service providers. This extensibility and consistency trade-off could be an area for further research and experimentation.
The API layer should offer a good developer experience and support innovation.	▶ The users of the Rosalind APIs were developers from service providers. To support innovation, the API layer needed to offer a good developer experience. During the project, a simple sandbox was set up to provide basic technical documentation and guidance, allowing users to raise issues and give feedback. The sandbox was implemented iteratively, with input from many user teams.

3.4 API functionalities

For the core API layer, the project team developed 33 API endpoints in six functional categories. A short description of these categories is provided below, with further details on all of the endpoints and their functionalities in Appendix 1. During the project, 541,865 API calls, 7,652 notifications and 1,595 errors were made, sent and logged. The project also implemented a CBDC with four decimal places to enable micropayments.

- **Account management (11 APIs):** Enables individuals and businesses to manage their CBDC accounts, eg open a new account, disable an existing account and check account balances. In the event that any CBDC access devices (eg phones or smart cards) are lost, a user could request to stop making payments by freezing his or her CBDC account temporarily. The user would still be able to receive payments. Account management APIs also include aliases. The use of aliases could improve interoperability with existing payments infrastructures, support privacy and empower user control over their personal data.

- **Payments (six APIs):** Enables a push payment initiated by the payer to one or more recipients. To ensure that payers are provided with full control over their payment transactions, pull payments are not supported. Instead, the Rosalind APIs support request to pay (RTP) and authenticated request to pay (ARTP). RTP requires the recipient to make a pay request to the payer, and the payer would have to specifically approve that request – by interacting through their PIPs – before a push payment could take place.

ARTP contains an authentication payload so that the recipient can send his or her payment request with the payload – via the recipient's PIP – to the payer's PIP for validation. This makes it possible for the request to be approved automatically and a push payment to be triggered. ARTP could support third-party payment initiation, for example for ESIPs. It could also support point-of-sale (POS) transactions, in which the recipient sometimes has internet connection, but the payer does not.

"During the project, 541,865 API calls, 7,652 notifications and 1,595 errors were made, sent and logged"

- **Programmability (10 APIs):** Aiming to provide a basic set of programmable features, the project developed three types of locking mechanism, namely two-party lock, three-party lock and hash timelock contract (HTLC) lock. The two-party lock allows the recipient to specify an amount to be reserved until a set of conditions are met for the payment to take place. The conditions would be agreed in advance with the payer. The recipient's PIP would be responsible for determining whether the conditions are met. This lock assumes a high level of trust between two transacting parties, and it may not work for all use cases. Therefore, the project added a three-party lock, which introduced a trusted third party that determines when the payment should be triggered. The HTLC lock would support atomic swaps and has the potential for integration with other systems. All locks have an expiry date and time to ensure that no CBDCs are locked indefinitely.
- **Participants (two APIs):** Enables service providers to obtain each other's public key for point-to-point encryption and to pull notifications after sending and receiving API calls.
- **ESIPs (two APIs):** This set of APIs enables individuals and businesses to "connect" and "disconnect" their CBDC accounts to and from third-party applications. This could support many use cases, such as third-party payment initiation, external smart contract applications and budgeting applications. Details on ESIPs can be found in Section 3.9 below.
- **Offline (two APIs):** Enables individuals and businesses to use CBDCs offline for offline payments and to bring CBDCs online later by transferring their balances.

3.5 Account- and token-based central bank ledgers

To assess whether the API could work with different central bank ledgers, the project tested simulations of both account- and token-based central bank ledgers. This approach demonstrated that almost identical API functionalities and endpoints would work for either ledger.

Some of the constraints in the account-based ledger, such as the inability to retrieve transaction history and user accounts, as well as the workaround for accessing lock information, also applied to the token-based ledger.

Two differences were identified between account- and token-based ledger structures. Firstly, the format of account numbers and service provider IDs was different, reflecting the differences in each ledger's underlying data model. Secondly, the offline payment solution tested during the project was Ethereum-specific, so it would not work for the token-based ledger.

Achieving almost identical API functionalities for each ledger structure demonstrated that it is feasible to abstract the ledger structure through a ledger-agnostic API while retaining compatibility for service providers and the wider ecosystem. It also showed the potential for the API layer to work with both account- and token-based ledgers. This is important because there are multiple reasons why different ledger technologies may be chosen, and many central banks are exploring and experimenting with a range of ledger designs and technologies for retail CBDC systems. It also showed that API design – which determines payments functionality – can be achieved before the design of the ledger, to ensure that user needs, rather than technical implementations, drive the solution.

"Achieving almost identical API functionalities demonstrated that it is feasible to abstract the ledger structure through a ledger-agnostic API, while retaining compatibility for service providers and the wider ecosystem."

3.6 Privacy model

The API layer assumed a privacy model that would not give the central bank ledger and API layer any visibility of or access to PII and payments data.

"Participants in the system – from either the public or private sector – should only have access to the minimum amount of information for them to play their defined roles and perform their specified tasks."

Only pseudonymous identifiers were created. User PII, together with other payments data and transaction history, was stored by service providers. Service providers' identities were pseudonymised to the central bank. Where necessary and with user permission, data could be shared between service providers (but not with the central bank) with end-to-end encryption when passing through the core API layer.

This privacy model assumed that a high level of privacy for end users is essential, but transactions would not be fully anonymous. Participants in the system – from either the public or private sector – should only have access to the minimum amount of information needed for them to play their defined roles and perform their specified tasks.

3.7 Security

It is essential that an API layer supports safe and secure retail payments. The project investigated and applied industry best practice to security features, such as authorisation, authentication, non-repudiation and anti-replay.

With regard to user authorisation, the Rosalind APIs explored the technological feasibility of allowing a PIP to act on behalf of individuals and businesses. Individual and business users are linked to a PIP, and only that PIP can be given the permission to make API calls (eg pay and lock funds) to pass on account owners' instructions to change account state and balance in the central bank ledger. If another PIP wanted to request a payment from an individual or a business, it would have to interact with the account owner's PIP. This was achieved by passing an alias to the lookup alias service, which returns the PIP ID for that account. In addition, the project explored the feasibility of allowing ESIPs to view account balances by connecting to the account in advance. This required that the appropriate permissions be granted to the ESIP by the account owner.[8]

To authenticate payment transactions, the project applied elements of the FAPI standard. Each API call was over HTTPS, using TLS1.2 and above. Authentication information must be included in the HTTP headers of all API calls. These headers are *x-fapi-financial-id* (ID of the service provider calling the API) and *x-api-key* (secret key of the service provider calling the API).

Non-repudiation is a security feature that provides assurance that a party cannot later deny having sent a message or made a payment. This is critical in disputes or fraud cases, as it ensures the integrity of payments. The Rosalind APIs used JSON Web Signatures (JWS) to sign and validate transactions for non-repudiation. The signatures used for signing transactions were required for all "write" API calls. For each API call, the signature was included in the x-jws-signature header of the message. During the project, this feature was turned off in order to accelerate use case development, but it would be a key security feature for any future production systems. The project also tested how this could apply to an offline use case in which the central bank would be asked to sign the transfer of funds from the user account to an escrow account held by the central bank. The signature would be used by an offline device to verify that the message came from the central bank and was intended for that particular device.

Implementing non-repudiation features could have performance implications. Signing messages and validating signatures could significantly increase the number of activities and the size of messages in the system, reducing speed and capacity to process transactions.

The final security feature explored was anti-replay. Replaying an API call requesting write access to the central bank ledger, either maliciously or accidentally, could result in an account being charged twice. The Rosalind APIs required each "write" API call to include a unique idempotency ID. An API call with the same idempotency ID as the previous one would not be accepted.

3.8 Standards

As a starting point for developing the APIs, the project explored how relevant industry standards could be implemented and whether any adaption was needed.

For example, the project tried to implement ISO 20022 messaging standards on a small number of APIs. This included *camt.103* for request to lock and lock APIs, *pain.013* for the RTP API and *pacs.008* for payment APIs. Modifications were made to ensure that these messages would work for the Rosalind APIs and were aligned for consistency. To reduce message size and increase interoperability, the project also used the ISO 20022 abbreviations for tag names. For authentication, the project applied FAPI standards (see Section 3.7). The use of the legal entity identifier (LEI) format was also implemented to help identify service providers.

3.9 Service providers

A CBDC ecosystem would benefit from diversity in service providers, through which new and creative ideas could be generated and turned into innovative products and services.

As such, the API would need to provide the relevant functionalities to support different types of service provider. The project experimented with API functionalities to support two possible types of service provider: PIPs and ESIPs. PIPs would be the gateways to a retail CBDC ecosystem. They would be expected to offer individuals and businesses digital "pass-through" wallets to manage their CBDC accounts and balances. They would also be expected to manage their relationships with these customers, execute their instructions to make payments, request others to pay and accept payment requests from others. They would be responsible for compliance with AML, know-your-customer (KYC) and CTF regulations. In the Rosalind APIs, PIPs are the default service provider type and would be able to use all APIs.

ESIPs would not provide wallets. They would specialise in other value-added services, such as providing tools for business analytics, budgeting and fraud monitoring. ESIPs would also work with PIPs to provide extra programmability services, for example by acting as decisioning parties in a three-party lock or as third parties to initiate payments. This is an area for further research and experimentation.

4 Use cases and user feedback

This chapter describes the use cases explored during the project in Section 4.1 and user feedback in Section 4.2.

4.1 Use cases explored

During the project, public and private sector collaborators developed more than 30 prototypes of retail CBDC use cases and tested the functionality, and demonstrated the capability of the Rosalind APIs. Key observations on the use cases are presented below, and a detailed list of those presented at the showcasing event and the TechSprint demo day event is set out in Appendix 2.

- The portfolio of use cases included a range of ideas, from addressing existing pain points in payments, enhancing user experience, and streamlining processes to enabling integration with existing payment rails, cards networks and POS interfaces. Some use cases explored new ways to make payments, and others tested applications with emerging technology. All investigated the potential for a CBDC system to support user needs in a more digitalised economy.
- The use cases covered a broad range of domains for individuals and businesses, such as peer-to-peer transfers, retail payments for goods and services, and small-value business transactions including receiving commissions, paying salaries and supporting trade finance.
- The use cases explored making payments online, in stores and offline, with the use of near-field communication (NFC), and via interactions with POS, QR codes, mobile phones, smart cards, biometric devices and smart assistants. The RTP and ARTP APIs have unlocked several interesting e-commerce solutions to streamline consumers' payments experiences.
- Some of the use cases also explored private sector programmability.[9] The API could enable individuals and businesses to put away a certain amount of CBDC for a specific use and to trigger payments under conditions that they have agreed in advance. The combination of locks and split payments has enabled a number of creative uses to make it possible for multiple payment legs to be settled at the same time.

  One use case also explored the use of rich content receipts, where consumers would be provided with both payment details and the conditions required for the consumer or merchant to trigger payments.
- Some use cases explored digital inclusion by testing a number of offline payments solutions. One use case explored digital inclusion through prototyping parent and child wallets. Solutions to enable micropayments were also developed. A few use cases tested how decentralised identifiers (DIDs) and verifiable credentials (VC) could be used to minimise the data stored in and transmitted through the central bank APIs for different types of payment, as well as how bank-verified user information could be used to fast-track the onboarding process.

4.2 User feedback and future improvement on Rosalind APIs

The exploration of use cases provided valuable feedback on the Rosalind APIs. Regarding functionality, user feedback suggested that the API was relatively simple and easy to use and provided robust handling of multiple payment legs. Some suggested that, to improve efficiency and performance, an asynchronous API model could be explored.

To support interoperability, the tiered architecture has the potential to enable retail payments services to operate seamlessly and to support more complex use cases. In addition, use cases could be developed in a number of ways on the API, offering choices to developers. Some users suggested that the APIs could benefit from further work to provide greater alignment with industry standards on security. Furthermore, as regards innovation and ecosystem building, the combined use of a few basic functionalities, such as split payments, lock and unlock and the four decimal places allowed in Rosalind, has the potential to support a number of potentially innovative use cases.

Within the project scope, the following items were identified as specific API functionalities that could benefit from future improvements:

- Regarding ESIPs, role-based rules and permissions could be implemented through the APIs to define the levels and types of access that ESIPs would need to have to serve user needs.
- The programmability functionality could be developed further to support conditional split payments; for example, when payment conditions are met, allowing multiple parties in a supply chain to be paid at the same time as goods or services are delivered. This could be achieved through enhancing the locking communication flow to enable individuals and businesses to reserve funds for and make payments to multiple recipients at the same time.
- Aliases and encryptions implemented in Rosalind were preliminary and would require significant development were they eventually to be deployed in a production system.
- In the Rosalind APIs, the HTLC lock was implemented with a standard SHA256 hash algorithm and an encoded 32 bytes hex value for secrets. This functionality could be expanded to offer different hash algorithms and length of secrets.
- The project focused on designing positive flows assuming a smooth completion of a CBDC payment transaction. This would not be sufficient for a payments system in which negative flows or "unhappy path" – setting out error states, alternative paths, and recovery journey – are critical to a good user experience.
- The Rosalind Sandbox could be further improved by enhancing documentation, providing test scripts and multimedia content for developers.

5 Insights, learnings and areas for further exploration

The project provided the following insights and learnings:

- A well-designed API layer could facilitate retail payments in CBDC. This was demonstrated through the wide range of front-end use cases explored during the project.
- A set of simple and core API functionalities could support a broad and diverse range of use cases. It also has the potential to enable a robust ecosystem to foster innovation in products and services that could help meet the future needs of a more digitalised society.
- The APIs could work with different central bank ledger technologies. In the project, simulations of both account- and token-based central bank ledgers were tested, and almost identical API functionalities and endpoints would work for either of the ledgers.
- The design of the API layer must be consistent with and implement the requirements of the wider privacy model for a CBDC. This was fundamental to the design and build of the Rosalind APIs. The privacy model was decided at the beginning of the project and before any API design decisions were made.
- APIs can support offline payments in CBDC, but there are a range of challenges involved in delivering offline functionality. Offline payments would require robust security and anti-replay protections. The specific design tested in the project – transfer of CBDCs off and back onto the central bank ledger with significant security controls to prevent counterfeiting, double-spending or loss of funds – created a much tighter coupling between the offline payment device and the core ledger than other payments use cases tested during the project. This tight coupling would limit the Rosalind APIs' ability to support different types of offline payments solution. Further research and experiment would be needed in this area to improve the Rosalind APIs.

The project work also suggested areas that could be explored further:

- The Rosalind privacy model requires user data to be captured, stored and managed by each private sector service provider. It allows permissioned information to be shared between two service providers through peer-to-peer encryptions. This raised further considerations about how the API might allow the ecosystem to share user and payments data in a privacy-preserving way, subject to user permission.
- In terms of API design, there was a trade-off between extensibility and consistency. The project aimed to keep APIs as simple and standardised as possible and allowed service providers greater flexibility to build bespoke features on their end for their specific use cases. This approach would help to maximise extensibility and support innovation, but it might potentially reduce consistency in user experience. More consistency would require a higher level of standardisation in API functionalities and therefore less flexibility for service providers to add bespoke elements.
- As the project tested the technological feasibility of an API layer capable of connecting systems, coordinating activities and facilitating payments, the need to define the operational roles and responsibilities of all participants in the ecosystem was identified. For example, the project explored how individuals and businesses could have multiple service providers linked to the same account and how they may be able to switch service providers quickly and easily. Whilst an important function, this type of arrangement could bring complexity. For example, there were questions about who the default recipient service provider might be, which service provider(s) should be trusted to initiate transactions on behalf of the user, and when things go wrong, which provider should take the responsibility.

6 Conclusion

Project Rosalind demonstrated that a well-designed API layer can enable a central bank ledger to interact with private sector service providers to safely provide retail CBDC payments.

The API layer can be ledger-agnostic so that central bank ledger choices can be abstracted. The API layer can also be application-agnostic, and a set of core API functionalities can support a broad and diverse range of use cases. Through collaboration with the ecosystem, the project has also demonstrated the potential of a CBDC system to enable a robust ecosystem to foster innovation, and to help meet the future needs of a more digitalised society.

This experiment has provided insights into and learnings on many important aspects of a retail CBDC system, such as API design, privacy, account- and token-based ledgers, security, standards, interoperability, programmability and ecosystem roles and responsibilities.

Endnotes

1 The six categories and 33 API endpoints are: Account (Open, OpenSubAccount, Disable, Enable, Freeze, Close, Alias, DeleteAlias, LookUpAlias, Balances and AvailableBalances), Payments (Pay, SplitPay, RequestToPay, AuthenticatedRequestToPay, Fund and Defund), Programmability (RequestToLock, TwoParty, ThreeParty, HTLC, CancelLock, DrawDownLock, DrawDownHTLC, LockbyLockID, LockbyPIP and LockbyAccount), Participants (Key and Notification), ESIPs (ConnectAccount and DisconnectAccount) and Offline (Download and Upload). See Appendix 1 for details.

2 See R Auer, G Cornelli and J Frost, "Rise of the central bank digital currencies: drivers, approaches and technologies", BIS Working Papers, no 880, 23 January 2023 (www.bis.org/publ/work880.htm) for the latest developments on retail CBDC around the world.

3 See BIS Innovation Hub, "BIS Innovation Hub work on central bank digital currency (CBDC)" (www.bis.org/about/bisih/topics/cbdc.htm) for details.

4 See Bank for International Settlements, *Annual Economic Report,* June 2021 (www.bis.org/publ/arpdf/ar2021e.pdf) for details on these options and Auer et al (2023) for the latest developments on retail CBDC around the world.

5 See Financial Stability Board, *Enhancing cross-border payments: Stage 3 roadmap,* 13 October 2020, www.fsb.org/2020/10/enhancing-cross-border-payments-stage-3-roadmap/

6 See BIS Committee on Payments and Market Infrastructures, *Interlinking payment systems and the role of application programming interfaces: a framework for cross-border payments,* 8 July 2022, www.bis.org/cpmi/publ/d205.htm

7 Due to the experimental nature of the project, these assumptions and design decisions should not be taken as recommendations or endorsements of any specific CBDC design choices.

8 The Rosalind APIs do not support non-custodial wallets, which are wallets that give owners exclusive control over private keys.

9 The Rosalind APIs do not support central bank-initiated programmability.

Appendix 1: Rosalind API endpoints and functionalities

Categories	Sub-categories	API endpoints	Description
Account	**Account management**	Open	Creates a new parent account on central bank ledger. Types could be personal and business.
		OpenSubAccount	Creates a new sub-account to a specific parent account on central bank ledger. Types could be personal and business.
		Disable	Enables the account holder to disable a parent or sub-account on central bank ledger. Once disabled, no activities will be allowed on this account.
		Enable	Enables a previously disabled parent or sub-account on central bank ledger.
		Freeze	Allows the account holder to freeze a parent or sub-account on central bank ledger. Depositing into this account is allowed, but withdrawing or making payments are not allowed.
		Close	Closes an account.
	Alias	Alias	Creates an alias on an account.
		DeleteAlias	Deletes (logically) an alias on an account.
		LookUpAlias	Returns details of an alias on an account.
	Balances	Balances	Returns the total balances of an account.
		AvailableBalances	Returns the total and available (not locked) balances of an account.
Payments	**Push payments**	Pay	Transfers CBDCs from one account to another.
		SplitPay	Transfers CBDCs from one account to multiple accounts.
	Request to pay	RequestToPay	Requests other accounts to pay.
		Authenticated RequestToPay	Enables the recipient's PIP to include an authentication packet with the RequestToPay so that the payer's PIP can automatically approve the request (i.e. for POS).
	Fund and defund	Fund	Adds CBDCs to an account.
		Defund	Draws down CBDCs from an account.

Categories	Sub-categories	API endpoints	Description
Programmability	**Set locks**	RequestToLock	Sends requests to lock funds in an account with one of the three types of lock specified below.
		TwoParty	Locks an amount of CBDC in an account. Decision to unlock and release the funds is given to the recipient's PIP. This lock contains an expiry date to ensure that funds will not be locked indefinitely.
		ThreeParty	Locks an amount of CBDC in an account. Decision to unlock and release the lock is given to a third-party PIP with an appropriate permission. This lock contains an expiry date to ensure that funds will not be locked indefinitely.
		HTLC	Locks an amount of CBDC in an account using HTLC.
	Cancel locks	CancelLock	Removes the lock previously placed on an account.
		DrawDownLock	Removes the TwoParty or ThreeParty lock previously placed on an account and draws down the funds either in full or in part.
		DrawDownHTLC	Removes the HTLC lock previously placed on an account and draws down the funds either in full or in part.
	Locks information	LockbyLockID	Returns information on a single active lock.
		LockbyPIP	Returns information on one or more active locks placed on all accounts with a specific PIP.
		LockbyAccount	Returns information on one or more active locks placed on an account.
Participants		Key	Returns the public key of a specific PIP. The key is used to send secure data between PIPs.
		Notification	Pulls notifications via the API. Webhooks with the same standards are also provided.
ESIPs	**Connectivity**	ConnectAccount	Connects an account to a third-party application or a merchant.
		DisconnectAccount	Disconnects an account from a third-party application or a merchant. This can be called by either the account holder or the connected party.
Offline	**Download and upload**	Download	Draws down CBDCs from an account holder's online wallet and adds the CBDCs to the account holder's offline wallet.
		Upload	Draws down CBDCs from an account holder's offline wallet and adds the CBDCs to the account holder's online wallet.

Appendix 2: Use cases explored in Project Rosalind

During the project, API user teams and TechSprint participating teams developed their front-end solutions and demonstrated the technological feasibility of their use cases by using the Rosalind APIs.

Use cases presented at the showcasing event are listed below:

- UC1 – a set of four retail CBDC solutions. The first solution demonstrated that during online checkout, a consumer could scan the merchant's QR code to pay. The second solution showed that during online checkout, a consumer could approve a payment request from the merchant. The third solution explored how, by allowing the merchant to scan a consumer's QR code, a payment request from the merchant could be accepted without further user approval. The last solution showed how a consumer could receive and store electronic receipts.
- UC2 – a prototype of parent and child wallets. Through interacting with the wallets, a parent could guide his or her child's digital payment experience and teach the child about earning money and responsible spending. This use case demonstrated the use of collaborative payments to enable cognitive accessibility.
- UC3 – a prototype that demonstrated how CBDCs could be reserved at the time of purchase and released on physical delivery of goods. It also demonstrated how CBDCs could interoperate with commercial bank money through an ecosystem services layer.
- UC4 – a prototype of an offline ledger connecting the Rosalind APIs that would make it possible to carry out consecutive offline transactions with instant and final settlements.
- UC5 – a prototype that would enable support for the use of CBDCs in the existing cards network, so that individuals could pay with CBDCs when travelling in a foreign country.
- UC6 – prototypes of two micropayments solutions. The first solution demonstrated how CBDC micropayments could support a corporate sustainability strategy. The second solution was a parking app that makes it possible to charge individuals for their parking by the minute.

Use cases presented at the TechSprint demo day event are listed below:

- UC1 – a prototype that demonstrated how consumers could link their wallets to merchants to buy subscriptions and make bill payments with "one-click checkouts", as well as how a CBDC wallet could interact with a smart assistant to enable voice-authenticated payments.
- UC2 – a use case that would allow governments to provide dynamic and real-time support to citizens on their energy bills.
- UC3 – a use case that would make it possible to accumulate points when making offline CBDC payments. These points could be used to support a set of charitable projects, and the donors would be rewarded with claimable non-fungible tokens (NFTs). This use case could help companies meet their social responsibility commitments.
- UC4 – a prototype that demonstrated how decentralised identifiers (DIDs) and verifiable credentials (VC) could interact with Rosalind APIs to enable users to discover and transact in CBDC securely, and with PII protected.
- UC5 – a prototype of a trade finance use case that demonstrated how a secure three-party escrow could be used to enable importers to lock the full price of the delivery and, upon receipt of goods, have it automatically released. This use case also showed how a marketplace of locked CBDC receivables could be accessed by potential investors.
- UC6 – a prototype of a CBDC wallet that demonstrated how, through open banking, individuals could use personal information verified by their banks to open CBDC accounts and make payments.
- UC7 – a use case that demonstrated how CBDCs could be used to improve the efficiency of payments for the itinerant or gig economy, where the employer and the worker could agree to programmable payments to be released upon completion of the work.
- UC8 – a use case that makes it possible for commuters to purchase train tickets using CBDCs and to be refunded immediately if the train arrives late. The refund process would be visible to all parties involved.
- UC9 – a set of four use cases that demonstrated how a traveller in a foreign country could use an app to set up an e-SIM on his or her phone to make secure payments, both online and offline, and to securely store CBDCs with a time limit to spend the funds.
- UC10 – a proof of concept that allows a user to load and unload CBDCs onto and from their offline smart-card device and their online CBDC wallet. Counterfeiting or double-spending could be monitored through the use of an offline ledger.
- UC11 – a use case in which the Rosalind APIs could facilitate saving and borrowing activities within a defined group chosen by individual users.
- UC12 – a use case demonstrating how a CBDC digital wallet could integrate with the existing POS interface to enable payments in shops and support merchant adoption.

Appendix 3:
Project participants, collaborators, and acknowledgement

Project sponsors

Francesca Hopwood Road, Centre Head, BIS Innovation Hub London Centre

Tom Mutton, Director, CBDC, Bank of England

Project team

Amy Jiang, Adviser, BIS Innovation Hub London Centre

Danny Russell, Principal Architect, CBDC, Bank of England

Carmen Barandela, Lead Architect, CBDC, Bank of England (until end of December 2022)

John Yeo, Adviser, BIS Innovation Hub London Centre

Vendor team

UST

Quant

API users group (in alphabetical order)

Amazon

Bank of Canada

Barclays

IDEMIA

Mastercard

Advisers group (in alphabetical order)

Richard G Brown, Chief Technology Officer, R3

Paul Carey, Staff Engineer, Stripe

Mitch Cohen, Chief Security Officer, eCurrency

Adrian Field, Director of Market Development, OneID

JJ Geewax, Principal Engineer, Google

Catherine Gu, Head of CBDC and Protocols, Visa

TechSprint participants (in alphabetical order)

Amazon

BMO

Boom

Budapest University of Technology and Economics

eCora

Global Cloud Payments

IDEMIA

Knox Networks

Magyar Nemzeti Bank (Central Bank of Hungary)

Millicent Labs

Nuggets

OneID

OneStep Financial

Revolut

SUPER HOW?

Secretarium

Thales

Vayana Network

Worldline

Acknowledgements

Special thanks to Cecilia Skingsley, Raphael Auer, Simon Scorer, Shantel Mullings and Codruta Boar for their inputs to this report; Beju Shah, Alan Soughley, Ben Dovey, Katie Fortune, Rachel Greener, Amy Lee, Ketul Patel and Ben Dyson for their support during the project; Eryk Walczak and Agnes Kennedy for supporting the project work within the London Centre; and Charlotte Crosswell, Markos Zachariadis, Morten Bech and Silvia Attanasio for the excellent panel discussion at the showcasing event.

We thank colleagues in the central banking community, individuals, teams, companies and organisations in the ecosystem that have offered their support, expressed their interest in participating in the project, attended the showcasing event and the TechSprint demo day event, and provided us with their valuable feedback.

BIS Innovation Hub

Bank of England

Project Rosalind
Building API prototypes for retail CBDC ecosystem innovation

bis.org