Committee on Payments and Market Infrastructures

BIS

Report to the G20

Promoting the harmonisation of application programming interfaces to enhance cross-border payments: recommendations and toolkit

October 2024

www.bis.org
email@bis.org

Follow us

Table of contents

Executive summary

The electronic exchange of payment data is vital to contemporary commerce and finance. Application programming interfaces (APIs) represent the latest in a series of technological advancements – beginning with the telegraph and extending to financial messaging networks like SWIFT – that facilitate such electronic data exchange. APIs are increasingly used throughout the global financial system for various payment functions, either supplementing or replacing traditional financial messaging to enhance the request and response functions in data exchange. The real-time automation offered by APIs greatly improves the efficiency of the request and response function of data exchange, especially as data requirements increase in scope and complexity. These features are especially pertinent for cross-border payments, where real-time processes can help verify the accuracy and completeness of payment data prior to execution, thus reducing downstream friction, delays and costs. As a result, integrating APIs into cross-border payments can produce considerable efficiency improvements in payment initiation, back office processes, payment tracking and status, reconciliation and reporting, among other functions.

However, the potential of APIs is significantly constrained by the current fragmentation of technical API standards. The lack of uniformity in these standards requires users of multiple services to invest in translation between different API standards, leading to increased development and processing times and costs as well as the risk of mistranslation. A variety of factors, including technological evolution, use of legacy systems, diversity of industry applications and use cases, and the absence of more centralised governance and coordination, contribute to the fragmentation of API standards.

The G20 has prioritised the harmonisation of APIs to improve the safety and efficiency of cross-border payments. The CPMI, in collaboration with the industry, has committed to formulate recommendations to promote greater API harmonisation in cross-border payments. This report presents the recommendations of a CPMI-led expert panel with industry, the API Panel of Experts (APEX). The recommendations are directed to a broad array of stakeholders involved in API harmonisation. However, they are also specific and limited, focusing on the recommendations deemed by APEX to be practical and capable of resulting in concrete benefits. The panel acknowledges that, while existing initiatives and data standards provide a solid foundation, the global API harmonisation process is less advanced than that for financial data exchange over messaging networks. Moreover, a higher level of heterogeneity may be justified in light of the diversity of API use cases. Therefore, the panel has neither attempted to develop nor recommended a single universal API standard for cross-border payments. The resulting recommendations in this report are neither binding nor prescriptive regarding specific technologies or API standards.

The recommendations are divided into four categories: i) recommendations that aim at facilitating the global API harmonisation processes; ii) recommendations that focus on API design principles and the use of existing international data standards; iii) recommendations to enhance the developer experience; and iv) recommendations to promote pre-validation APIs and implementation. Each recommendation is accompanied by a list of potential action items that stakeholders may consider as practical and concrete implementation measures for the recommendation. The recommendations are further supported by a toolkit (Annex 1) to assist various stakeholders in assessing their current practices in relation to the recommendations.

API harmonisation for cross-border payments will be a lengthy process involving a diverse range of stakeholders. It is therefore recommended that progress on monitoring the suggested action items for each recommendation remain a priority for authorities and private stakeholders.

1. Introduction

Enhancing cross-border payments has the potential to offer widespread benefits, through lower costs, faster speed, greater transparency and improved access. Since October 2020, when G20 leaders endorsed the roadmap for enhancing cross-border payments (FSB (2020)), the CPMI, in coordination with the Financial Stability Board (FSB) and other relevant international organisations and standard-setting bodies, has largely laid the foundation for further developments through stocktakes and analysis. Regular collaboration with a wide range of private and public sector stakeholders who share their insights and expertise has also contributed to a better understanding of the challenges. In February 2023, the FSB published the prioritised roadmap to enhance cross-border payments; this CPMI report completes one of the priority actions of the cross-border payments programme implementation (FSB (2023)). It provides recommendations for the harmonisation of application programming interfaces (APIs) for enhancing cross-border payments.

The real-time automation offered by APIs greatly improves the efficiency of the request and response function of data exchange, especially as data requirements increase in scope and complexity. These features are especially relevant for cross-border payments, where real-time processes can verify that payment data are correct and complete before execution, minimising downstream friction and reducing delays and costs.

Recognising the benefits of adopting APIs in payments, financial market infrastructures (FMIs) and payment service providers (PSPs) are increasingly using APIs globally to supplement or even substitute for traditional financial messaging in a variety of payment functions. According to a recent CPMI survey, 45 (or 65% of) RTGS systems and 42 (or 93% of) fast payment systems (FPS) plan to use APIs within the next five years. Existing use cases include submitting or amending payment instructions, accessing transaction/statement data, and receiving/managing transaction notifications (Fitzgerald et al (2024)).

While APIs hold potential for enhancing cross-border payments, the current wide diversity in how they are developed, implemented and used is substantially limiting this potential. This diversity forces users of multiple services to invest in translating between different API standards, increasing processing and development times, costs and the risks of mistranslation. A more harmonised approach in how APIs are designed, implemented and used would enable and facilitate the straight through data processing and automation enhancements that APIs promise. One particularly beneficial example is in the context of FPS interlinking (CPMI (2022)). In a multilateral hub-and-spoke model of FPS interlinking, FPS operators would be required to implement only one harmonised set of APIs rather than customised APIs for each bilateral connection between payment systems.

Notwithstanding the fragmentation, a range of international payment and data standards provide useful building blocks for more harmonised APIs which may not be self-evident to the API developer community. API harmonisation thus partly rests on supporting efforts to build on these standards and not necessarily create new ones. At the same time, the diversity of use cases, incentives and markets, as well as the early stage of harmonisation initiatives – especially at the global level – suggest that expectations regarding the scope of harmonisation should be modest. In this vein, promoting robust API standards that are voluntary, open and consensus-based may be more effective in achieving API harmonisation over the long run than aiming for a single universal API standard that would have weak prospects for adoption.

This report thus sets out 10 recommendations that encourage leveraging existing standards where currently underutilised, but also recognises the legitimacy of diverse technical approaches based on diverse use cases. All recommendations contain potential action items that are specific and practical and that target various stakeholders involved in payments-focused API harmonisation.

1.1 What are APIs and their potential benefits for cross-border payments?

1.1.1 Basic function and value-enabling properties

APIs provide the means for a software application (eg a mobile phone application) to request a specific piece of data from one or more other software applications (eg a core banking platform installed on a server in a financial institution's data centre). From there the data can be transferred from the data-providing application(s) back to the requester, provided the original request was valid.[1] Put another way, APIs are the "messenger" offering to carry an instruction from the requester to the provider of the information and return with the answer (Graph 1). In doing so, the API specifies the required rules, language and vocabulary for those requests to ensure that the receiver of the request can always quickly, easily and accurately understand the request and provide the answer in a way which the requester can immediately understand.[2] The API also specifies the security protocol to assure the receiver of the request that the request is authentic and legitimate. The requesting and responding systems do not need to understand how the other systems work, with the API interface being the only area of standardisation. As a result, this approach reduces the need for bespoke system-to-system communication arrangements. Importantly, often no direct user intervention is required to initiate or respond to individual information requests; instead, such requests are automatically initiated as part of the broader functionalities invoked by end users.[3] This feature greatly improves the efficiency of the request and response function of data exchange, particularly as data requirements increase in scope and complexity.

By facilitating communication between software applications, APIs enable the applications to expand their own functionality by leveraging that of other applications. Applications using APIs do not need to replicate the functionality or services already provided by other applications, but simply need to be able to interface with them. The ability to invoke and leverage the capabilities or data of other applications through APIs is a key driver of value creation, new markets and enhanced user experience in the digital economy. In this way, APIs can be likened to the connective studs of Lego bricks, which enable the creation of structures of various formations depending on the creativity of the builder and the specific needs and preferences of users.[4]

1.1.2 Potential benefits of APIs in cross-border payments

Several areas of the cross-border payments processing chain may be enhanced by the greater automation, efficiency, transparency, customer experience and innovation afforded by APIs.

- *Payment initiation:* APIs may facilitate payment initiation, reducing the number of manual steps for end users to initiate cross-border payments with their PSPs. APIs can also provide real-time

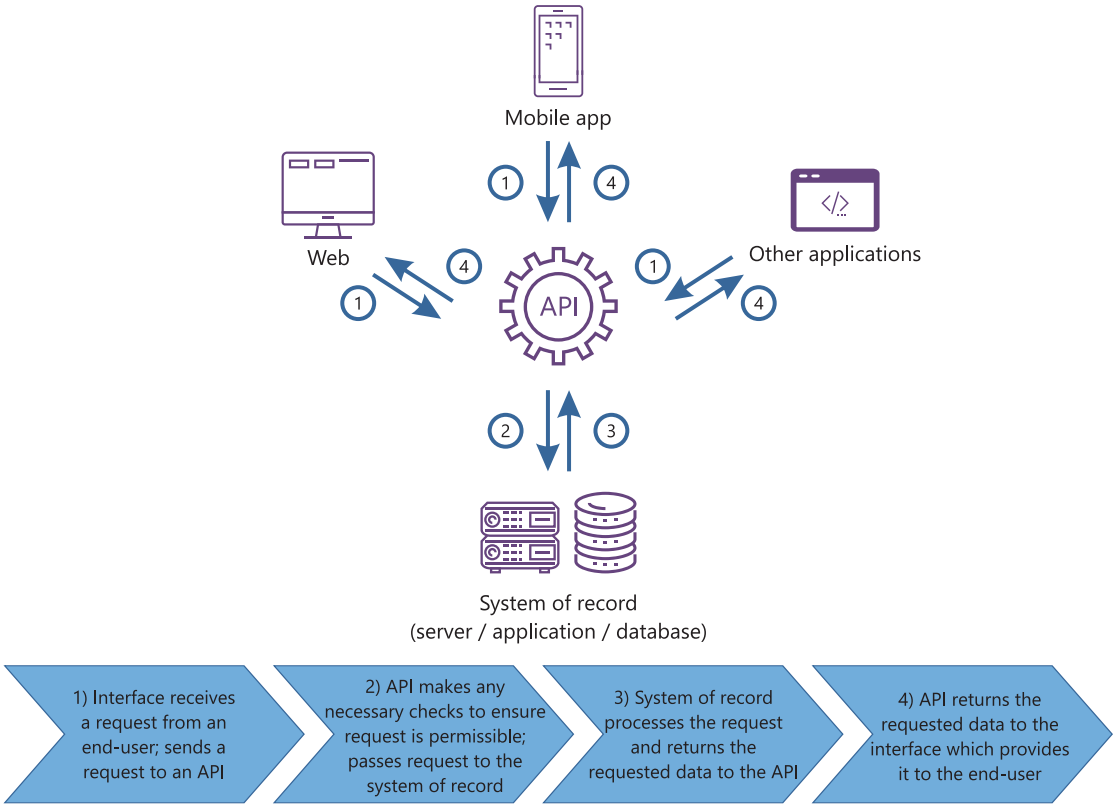
[1] When an API service and two software applications are on different hardware platforms, a communications network must link the three services (eg either a closed network, such as a corporate network, or the internet). APIs that communicate over a network are also called "web services" (even though some APIs are only used to interface between internal applications). The development of the predecessors to modern-day APIs therefore has a relatively long history and can be traced back to uses of linked databases in the 1950s. These first connections were initially between pairs of linked computers and then increasingly small, local, closed networks.

[2] In technical parlance, API rules are "protocols" (eg HTTP, SOAP, REST, etc), languages are "formats" (eg XML and JSON) and vocabularies are "data dictionaries" within a messaging standard (eg ISO 20022). For further details, please refer to the Glossary.

[3] An example would be usage of services embedded within external applications, such as when an individual can access information from a ride-hailing app directly from a navigation app without the need to separately access the ride-hailing app.

[4] While the data exchange objectives are similar, financial messaging systems differ from APIs in that they require a messaging interface to intermediate requests between internal and external systems. This intermediation requires a stricter adherence to internal workflow processes and schedules for data exchange. By contrast, APIs provide the potential for the internal systems of different entities to be more directly connected and may trigger actions within the systems they are interacting with across a wider range of functions, facilitating more real-time data exchange.

Example of basic functioning of APIs Graph 1



Source: CPMI.

exchange rate information, allowing end users to calculate currency conversion costs more accurately and helping them make informed decisions based on up-to-date market rates.

- *Back office processes:* APIs may be used for back office, screening and compliance processes involved in a traditional correspondent banking chain. Four areas in particular stand to benefit from APIs. First, APIs may enable pre-validation of payments prior to the sending of cross-border payment messages, increasing the likelihood of successful settlement and limiting the number of costly rejections along the correspondent banking chain. Second, anti-money laundering (AML), sanctions screening and know-your-customer (KYC) verifications could be executed in real time, facilitated by centralised and API-based screening services offered by non-banks.[5] Third, APIs can integrate with fraud detection systems to identify and prevent fraudulent cross-border transactions. Fourth, they may also support more efficient message repairs and investigations (eg requests for further information) which today are conducted with respect to the next intermediary bank in the correspondent banking chain. The linear passing of investigation requests can be time-consuming as well as costly and is a major source of delays in a cross-border payment. APIs create the possibility of more direct, automated and standardised data exchanges between the originating and intermediate banks related to message repairs and investigations.

[5] APIs may more generally facilitate the emergence of a financial ecosystem in which third parties can help banks improve the efficiency of their back office and compliance processes as well as expand access to cross-border payments. For example, dedicated API data links with external data providers may improve the efficiency of both compliance investigations (eg to retrieve background information on relevant parties in the payment chain) and access to externally provided machine learning or artificial intelligence solutions.

- *Payment tracking and status:* APIs can support enhanced payment tracking and status updates. They can provide real-time updates on the status of cross-border payments, allowing end users to track the progress of their transactions, improving transparency and speed through more efficient resolution of message investigations.
- *Payment reconciliation and reporting:* APIs can facilitate the automatic reconciliation of cross-border payment data with internal systems, streamlining the accounting process and providing end users with more accurate and timely financial reports and statements.
- *Product and service innovation:* APIs promote collaboration between different stakeholders in the cross-border payment ecosystem. Financial institutions, PSPs and other entities can connect their systems and services through APIs, enabling automated data exchange and interoperability. This collaboration fosters innovation, expands market reach, and enables the development of new payment solutions.
- *Interlinking of payment systems:* Aside from making back office processes and compliance screening more efficient, APIs could facilitate linking between payment systems in different jurisdictions, potentially bypassing or reducing the length of traditional correspondent banking chains. In particular, using APIs to link domestic FPS could dramatically increase the speed of cross-border payments relative to payments on traditional rails.

1.2 Fragmentation of API standards and role of harmonisation

While use of APIs in cross-border payments can bring benefits, there is no universally agreed upon standard. Lack of an API harmonisation approach is a major hurdle to their consistent and fast adoption. The fragmentation or even lack of API standards has been cited by stakeholders as the biggest challenge to wider adoption. Fragmentation of standards creates frictions for the party using multiple services, requiring them to invest in translation between different API formats (eg developing separate APIs for each third-party interface). This fragmentation increases processing and development times as well as costs and introduces additional risks (eg mistranslation).

1.2.1 Reasons for fragmentation of API standards

A number of reasons account for the current fragmentation in API standards:

- *Evolution of technology and legacy systems:* The rapid evolution of technology and the emergence of new platforms, frameworks and protocols have contributed to a proliferation of new API standards. Older APIs may still use outdated standards because updating them could break existing applications that rely on them, contributing to a fragmented landscape. Moreover, many organisations have legacy systems that may not be easily compatible with modern API standards. In these cases, multiple standards can coexist as organisations continue to support and maintain their legacy systems alongside newer API standards.

  Industry diversity: The very wide range of API use cases also contributes to fragmentation in API standards. Various industries have different requirements and use cases for APIs. API services have typically been established by different providers at different times, reflecting different stages of digitisation, legacy technology, and cost/resource considerations. Users will want to connect to a wide range of API services, with each fulfilling different functions according to their specific business needs. They may want to use certain services to retrieve pieces of information (eg from a range of suppliers across different industries) while at the same time supplying different, though related, information to customers or data requesters.
- *Competitive landscape:* Competition among technology providers and PSPs can also contribute to fragmentation. Each organisation may develop its own proprietary APIs to differentiate itself in the market, resulting in a fragmented landscape of competing standards.

Box 1

What is the difference between API standards and API harmonisation?

Standardisation and harmonisation are two terms often used in the context of improving processes, systems and interfaces, including APIs. While they share common elements, they have distinct meanings.

API standards

API standards are configurations of technical specifications related to an API's rules (protocols), languages (formats), data dictionaries and security features that find a degree of common adoption in industry. The aim of API standardisation is to ensure that certain processes, materials, or interfaces are consistent and can be universally applied. Standards are typically established by recognised standardisation bodies and are often voluntary, although they can be made mandatory by law or contract. In the context of APIs, standardisation might involve adopting widely accepted protocols and conventions like REST or SOAP, or data formats like JSON or XML (see Glossary). API standards may be market-driven, with market actors converging on a particular configuration of technical specifications over time of their own accord, or arise through a process facilitated by authorities or technical standard-setting bodies. Despite the long-standing use of APIs in a variety of contexts, standardisation of APIs is relatively new due to a number of factors related to an inherent tendency for fragmentation. In the context of open banking initiatives, authorities and industry associations in several jurisdictions have launched efforts to facilitate the development of API standards (see Annex 2). However, at present no overarching global API standardisation body exists.[6]

Implementation of API standards may not always be regulated by the same bodies that formulated the standards. For example, ISO is a standards organisation but is not able to mandate the implementation of drafted standards. For some standards organisations, the distinction between standardisation, implementation and scheme activities is a crucial guiding principle. It enables them to focus on technical and organisational requirements (unaffected by scheme and business arrangements, and not being dominated by market implementers). Implementation aspects are then only covered to the extent that they help to identify and manage interoperability or technical implementation issues related to the standards.

API harmonisation

API harmonisation refers to the process of ensuring that multiple APIs, often across different systems or platforms, follow a consistent design and structure. Harmonisation is particularly important in large systems where multiple APIs are used, as it can significantly reduce the learning curve for developers and increase the speed and efficiency of development and integration processes. The goal is to make these APIs more interoperable and easier to use and understand, and to reduce the complexity of integration.

In essence, while standardisation is about setting a common set of rules or guidelines, harmonisation is about aligning different systems or processes to work together seamlessly, as well as aligning on the most important data. Standardisation can be a part of the harmonisation process, as adopting standards can help achieve harmonisation. But harmonisation often involves additional steps, such as aligning design principles, error handling, security measures and more.

Source: CPMI.

- *Regulatory and market practice differences:* Entities adopting APIs may consider the needs of the domestic market rather than practices that are common in other jurisdictions. APIs may also be designed to meet the requirements of a particular piece of jurisdictional legislation or regulation. This diversity of factors requires a flexible and adaptable approach towards the formulation and application of API standards. Moreover, standards are not static but evolve in response to shifts in market dynamics and changes in regulatory landscapes. Hence, ongoing engagement with

[6] The International Organization for Standardization (ISO) has established an API Standardisation Working Group, but standardisation for APIs is at an earlier stage relative to financial messaging (eg ISO 20022).

these shifts and changes is recommended to ensure that the API standards remain effective and relevant.

- *Security considerations:* Different APIs may have different security requirements, which can lead to variations in the way APIs are designed and implemented.
- *Lack of centralised governance and coordination:* There is no single governing body that sets universal API standards. As a result, different organisations may develop their own standards independently, without considering existing standards or seeking broader alignment. This lack of coordination and collaboration among industry stakeholders contributes to the fragmentation of API standards.

It is important to acknowledge that while fragmentation can pose challenges, the diversity of API standards can also foster innovation, promote competition and enhance consumer choice. Moreover, it can also offer opportunities for smaller entities to participate and compete in the market. Implementers or hubs can still guarantee interoperability among a limited range of diverse API standards.

1.2.2 Benefits of API harmonisation for cross-border payments

Fragmentation of API standards can impede the adoption of APIs by making implementation more costly and time-consuming. Increased standardisation and harmonisation can work in the opposite direction: it can help to speed up adoption of APIs with potential benefits for the speed, cost, transparency and accessibility of cross-border payments. For example, it would boost the efficiency and usage of APIs in cross-border payments by utilising existing payment routes in designing and implementing, for example, pre-validation APIs. Greater harmonisation of API standards can also support the interlinking of FPS. In a hub-and-spoke interlinking model involving a platform that stands in the middle of links between payment systems, developers would need to develop one standardised set of APIs towards the hub rather than customised APIs for each bilateral connection between payment systems. This harmonisation could thus lead to significant reductions in development costs and time to market.

Greater standardisation of APIs may also yield access benefits. Cross-border payments today rely heavily on banks and their access to traditional messaging networks. APIs promise to facilitate technical access for actors that are qualified to process cross-border payments but do not have access to proprietary messaging networks.

1.3 CPMI's API Panel of Experts (APEX): collaboration with industry on developing recommendations for harmonisation of APIs for cross-border payments

The harmonisation of APIs for cross-border payments is a priority action of the G20 Roadmap. The CPMI, in conjunction with industry, has agreed to develop recommendations to encourage greater harmonisation of APIs in cross-border payments.[7] As in other areas of technical standard-setting, industry provides the expertise and knowledge of business and technology trends that are necessary for recommendations on potential API standards to be relevant and timely.

To help draft the recommendations, the CPMI drew upon an API panel of experts ("APEX"), with members drawn from central banks and industry. Established in late 2023, its members were selected to

[7] In particular, "CPMI to convene relevant stakeholders (including jurisdiction-level harmonisation bodies, data authorities, and international organisations) in a joint industry panel to (i) assist in the evaluation of proposals for a set of API standards that will utilise one or more of the (existing) protocols as appropriate to ensure interoperability of cross-border information exchange, (ii) develop recommendations for greater harmonisation of APIs used in cross-border payments, and (iii) develop a longer-term global governance proposal and process to continue updating harmonised API requirements." (FSB (2023)).

Box 2

Stakeholders in payments-focused API harmonisation

API harmonisation involves a diverse range of stakeholders whose perspectives should be considered as part of any harmonisation initiative for payments-focused APIs. Stakeholders include:

Payment service providers (PSPs): Banks, non-bank PSPs, and other financial institutions play a crucial role in API harmonisation efforts. They contribute their expertise and requirements to ensure that APIs meet the needs of the financial industry. Financial institutions also collaborate to define common standards and frameworks for secure and interoperable APIs.

Payment systems: Payment systems, both wholesale and retail, are increasingly adopting APIs for various payment functionalities. As a critical infrastructure for clearing and settlement of payments, payment systems may be highly influential in setting de facto or de jure API standards in their payment areas.

Jurisdictional authorities and regulatory bodies: Authorities, such as central banks and financial regulators (or collective bodies of such authorities), are important actors in API harmonisation. They provide guidance and establish regulations that govern the use of APIs in the financial sector. Regulatory bodies may work closely with industry stakeholders to develop standards and ensure compliance with regulatory requirements.

Standards organisations: Organisations that develop and maintain standards for APIs, such as the International Organization for Standardization (ISO) and the Financial Information eXchange (FIX) Protocol, are involved in API harmonisation. These organisations facilitate the development of industry-wide standards and promote interoperability among different systems and platforms.

Industry associations: Associations and consortiums focused on the financial industry, such as the Financial Data Exchange (FDX) and the Open Banking Initiative, play a significant role in API harmonisation. They bring together financial institutions, technology providers and other stakeholders to collaborate on defining common API standards, security protocols and data formats.

Technology providers: Companies that develop and provide technology solutions for the financial industry, including API management platforms, payment gateways and software vendors. These companies contribute their expertise and work towards aligning their products and services with industry standards to ensure seamless integration and interoperability.

Developers and integrators: Developers and system integrators who build applications and integrate systems using APIs can provide feedback and experience in working with APIs to help shape the standards and best practices. Developers also contribute to the development of software development kits (SDKs) and developer tools that facilitate API integration.

End users: The end users of financial services, such as businesses and consumers, indirectly influence API harmonisation. Their needs and expectations drive the demand for standardised APIs that offer seamless and secure access to financial services and data.

Source: CPMI.

ensure a balanced representation of central banks, payment system operators and other market infrastructures, jurisdiction-based API standardisation associations, and the private sector at large, reflecting a variety of business models (see Annex 3).

2. API harmonisation recommendations

2.1 Scope and structure

The recommendations in this section are designed to be capable of resulting in concrete benefits. They do this by directly addressing stakeholders and setting out potential actions while outlining reasonable constraints and boundaries in their implementation. Some recommendations focus on suggested best practices regarding API design, build, implementation and use (the "what"), while others focus on governance and the coordination and implementation process (the "how"). Each recommendation in this report can be linked to at least one of the G20's cross-border payments targets (Table 1). Moreover, all of the recommendations have a broad impact on an array of stakeholders in payments-focused API harmonisation (Box 2 and Table 2).

Underlying all of the recommendations is a recognition that the global API harmonisation process is at a relatively early stage compared with financial messaging. It will require continuous and close consultation between the public and private sectors and could be subject to change as the landscape evolves. The range of international payment and data standards already provides useful building blocks for more harmonised APIs. This may not be self-evident to stakeholders in API standard-setting. API harmonisation thus partly rests on supporting efforts to build on these existing standards. At the same time, the diversity of use cases, incentives and markets, as well as the early stage of harmonisation initiatives, should temper expectations regarding the feasible scope and speed of harmonisation. Greater harmonisation of APIs in cross-border payments cannot be imposed.

In this vein, it is important to clarify what the recommendations are *not*. They are not an attempt to develop and impose a single universal API standard or prescribe specific technologies for cross-border payments. This level of specificity would not be realistic; it would neither recognise market-specific needs, nor recognise that a large part of the process is expected to be industry-driven. The recommendations also do not attempt to cover API usages beyond those of cross-border payments. Moreover, the CPMI recognises that there are limits to the representativeness of the APEX and hence to the breadth of perspectives that could be expected of a group of this size. As such, these recommendations are intended to set forth a process of API harmonisation that is matched to the needs and level of maturity of the cross-border payments ecosystem.

The recommendations are grouped into four categories: i) recommendations intended to facilitate the global harmonisation processes; ii) recommendations on design principles and use of existing international data standards; iii) recommendations to enhance the developer experience; and iv) recommendations promoting pre-validation APIs and implementation. Each recommendation is followed by a list of potential action items that stakeholders may consider as practical and concrete implementation measures for the higher-level recommendations. The recommendations are also supported by a toolkit of checklists in Annex 1 to help various stakeholders evaluate their current practices with respect to the recommendations.

API harmonisation recommendations and their link to the G20 targets

Table 1

Recommendations	Link to G20 targets: Cost	Speed	Access	Transparency
I. Harmonisation process				
1. Open API standards *Standards that are voluntary, open and consensus-based promote the development of open and transparent payments. This also facilitates cheaper and more accessible payments.*				
2. Existing harmonisation initiatives *Aligning with existing harmonisation initiatives lowers barriers to entry for developers, encouraging greater participation and promoting competition and innovation.*				
II. Design principles and data standards				
3. Best practice design *Following API design best practices and reusing existing international standards lowers barriers to entry for developers, encouraging greater participation and promoting competition and innovation.*				
4. ISO 20022 and FATF Recommendation 16 *Aligning with global payment message consensus standards reduces the need for translation or truncating data, promoting interoperability resiliency and therefore lowering costs.*				
5. Security *Incorporating prevalent security standards reduces interoperability issues and injects greater clarity and confidence into the exchange of payment message data.*				
6. Common registries *Utilising common registries for identification lowers barriers to entry for developers, encouraging greater participation and promoting competition and innovation.*				
III. Developer experience				
7. Developer training and knowledge transfer *Facilitating knowledge-sharing and developer training lowers barriers to entry for developers, encouraging greater participation and promoting competition and innovation.*				
8. Developer resources *Producing resources for developers, including guides and tools, lowers barriers to entry for developers, encouraging greater participation and promoting competition and innovation.*				
IV. Value chain and implementation				
9. Pre-validation *The greater use of pre-validation developed for users by PSPs increases straight through processes, reducing the cost and delays involved in the payment while providing more information.*				
10. Progress tracking and promoting adoption*				

Dark green indicates that the recommendation has a major direct impact on the G20 target. Green indicates that the recommendation has a direct impact on the target. Light green indicates that the recommendation has an indirect impact on the target. Empty indicates no impact on the target.

*For recommendation 10, the impact assessments across the four targets reflect the maximum values assessed for recommendations 1–9.

Source: CPMI.

CPMI APEX recommendations: targeted stakeholders

Table 2

Recommendations	Stakeholders*							
	PSPs	Payment systems	Authorities	Standards orgs	Industry assocs	Tech providers	Developers	End users
I. Harmonisation process								
1. Open API standards								
2. Existing initiatives								
II. Design principles and data standards								
3. Best practice design								
4. ISO 20022/FATF Rec 16								
5. Security								
6. Common registries								
III. Developer experience								
7. Developer training								
8. Developer resources								
IV. Value chain and implementation								
9. Pre-validation								
10. Progress tracking								

Dark green indicates that the stakeholder is responsible for implementing the recommendation. Green indicates that the stakeholder is directly impacted by the recommendation. Light green indicates that the stakeholder may be impacted by the implementation of the recommendation. Empty indicates no material impact on the stakeholder.

*See Box 2 for definitions.

Source: CPMI.

2.2 Recommendations and potential action items

2.2.1 Recommendation 1: Open API standards

All stakeholders in API standardisation, but especially jurisdictional authorities and standards organisations, should actively support the development of cross-border payment API standards that are voluntary, open and consensus-based.

Successful API harmonisation rests on API standards that are future-proof and sustainable (ie those that meet a market need and are technically sound). The best assurance of future-proof, sustainable API standards is a scenario where there are open APIs[8] that have been arrived at through an open, consensus-based process and where adoption of the API standard is voluntary. While some standards are proprietary, with the intellectual property privately owned, communities are much more likely to adopt an open API

[8] Open APIs are publicly available and accessible to anyone who wants to use them, without requiring any authorisation or registration.

standard than a proprietary one, where there is always the risk that the owner of the intellectual property will seek to control or monetise the standard, or simply change it unilaterally. Voluntary adoption subjects API standards to an additional market relevance test.

APIs for cross-border payments are currently being designed in a dynamic market landscape where regulations, technology and market requirements are in constant flux, challenging efforts to harmonise and standardise APIs. The evolution of API standards in this environment requires a comprehensive understanding of the market's intricacies as well as strategic planning of potential adjustments. In light of the complexity of the dynamic market landscape, the participation of all stakeholders (including, in particular, API developers) in the decision-making process for API standards is critical. Broad-based insights and perspectives can provide valuable input regarding the potential implications of changes to existing standards. This collaborative approach ensures that the modifications are in the best interests of all parties involved, technically sound and market relevant. It also fosters a sense of ownership and commitment to the harmonisation process.

Potential action items:

1. All stakeholders should encourage API specifications to be published and maintained under the auspices of an existing open standards body (eg ISO, World Wide Web Consortium (W3C)) or take inspiration from these successful standards bodies for any new governance construct.
2. API standards organisations may consider drafting and supporting a cross-border payments API harmonisation charter by which stakeholders may publicly affirm their support for voluntary, open and consensus-based API standards, among other principles related to the process of API standardisation and harmonisation. As a starting point, API standards organisations may wish to review their alignment with the principles in the open governance checklist included in the toolkit (see Annex 1).
3. API standards organisations should encourage broad participation by stakeholders in the standard-setting processes (at the national, regional and/or global levels) to ensure that the global standards reflect the diverse perspectives and priorities of different jurisdictions. API developer perspectives, in particular, should be accommodated to the extent possible, acknowledging the possible need for quicker maintenance cycles and updates to baseline standards.

2.2.2 Recommendation 2: Existing harmonisation initiatives

Jurisdictional authorities (eg central banks, relevant government agencies, and regulatory bodies) and standards organisations should leverage the experience of existing API harmonisation initiatives that have acquired extensive experience and lessons in the harmonisation of diverse API standards. This should include enhanced cooperation and information-sharing across the relevant jurisdictional authorities and standards organisations.

While API harmonisation initiatives at a global level are at a relatively early stage, there are a range of jurisdiction- or use case-specific initiatives with experiences and lessons which may be leveraged for regional or global harmonisation efforts. Examples include India's UPI, the BIS Innovation Hub's Project Nexus, and the European Open Banking/Open Finance framework. These initiatives have acquired significant experience in harmonising diverse API standards, including grappling with challenges presented by, among other things, differences in governance, business and market models, functional models and technical architecture models.

Potential action items:

1. API standards organisations should engage in regular consultations, joint initiatives and sharing of best practices to enhance the effectiveness of their standards as well as to promote trust and cooperation among different entities. A starting point would be mutual transparency over motivations for harmonisation alongside identification of common challenges.

2. API standards organisations should develop regionally harmonised standards or guidelines that build upon and are consistent with international standards, while addressing region-specific concerns or priorities. These regionally harmonised standards or guidelines can serve as a stepping stone towards global harmonisation.

3. Jurisdictional authorities should engage in cooperation and information-sharing with respect to harmonisation goals and agendas, technical specifications, and regulatory aspects of API standards and harmonisation efforts.

2.2.3 Recommendation 3: Best practice design

API developers should follow the API best practice design principles laid out in this report in designing cross-border payment APIs. Doing so will support greater harmonisation of cross-border payment APIs, facilitating the adoption of APIs and the G20 goals of faster, cheaper, more accessible and more transparent cross-border payments.

The best practice design principles for APIs laid out in the toolkit (see Annex 1) are intended to help developers design cross-border payment APIs in a more harmonised direction. The design principles cover i) the reuse of international data standards; ii) business processes and context that should influence the design of cross-border payment APIs; and iii) guidelines around the technical implementation of APIs.

For example, API developers should reuse, whenever possible, predominant existing international standards for reference data, data semantics and standard data resource models. With respect to reference data, the use of such data newly generated ad hoc should be a last resort. With respect to data semantics, APIs should reuse standard business definitions and data types to ensure end-to-end interoperability and compatibility with existing infrastructure. And with regard to data resource models, for API designs that are intended for broad usage, API developers should consider using, and where necessary extending, standard data resource models. Implementers should seek to liaise with the submitter of the standard resource model (and the ISO 20022 registration authority in the case of ISO 20022) to propose new changes or new resources for future releases.

It is important for API developers to be aware of the broader context and business processes into which their APIs will fit. APIs for cross-border payments will in most cases be part of an end-to-end payment process involving other parties and markets using their own APIs. Reuse of existing international data standards is one way to support interoperability, but awareness of the broader business and multinational picture will help to avoid API design dead ends.

Finally, regarding technical implementation, API designs should conform to established industry design patterns and conventions, be implementation-agnostic, and be subject to a clear and consistent governance policy around versioning. The evolution of APIs and API standards is inevitable. It is important that no undue fragmentation is introduced in the evolution of API standards due to the absence of a strong versioning and maintenance framework.

Potential action items:

1. Jurisdictional authorities should distribute the API harmonisation design principles to payment system stakeholders (eg operators, PSPs, participants, API standards organisations) within their jurisdictions, especially those directly involved in the design of APIs.

2. Organisations engaged in developing cross-border payment APIs should initiate internal discussions to assess the potential for incorporating the API harmonisation design principles into their API designs.

3. All stakeholders could consider adopting and implementing the international standards issued by the standards organisations referenced in the API harmonisation design principles as a baseline, while allowing for reasonable adjustments to suit national or regional circumstances.

2.2.4 Recommendation 4: ISO 20022 and FATF Recommendation 16

API developers should adopt ISO 20022 semantics and resource models, to the extent practicable and appropriate for the given user community and use case, as this can significantly speed up API development and strengthen interoperability. Moreover, API developers should align with the harmonised data models and international regulatory guidance in determining data content for cross-border payment APIs to ensure consistency with guidelines on ISO 20022 market practice (eg CBPR+, HVPS+, the CPMI's ISO 20022 harmonised data model) and compliance with regulatory guidance on payment transparency (eg FATF Recommendation 16).

When seeking to harmonise APIs, there are strong reasons to ensure that the underlying data standard or model (ie the semantics of the formatting language) of an API data exchange is consistent with that used by existing payment messaging standards, in particular ISO 20022. Many payment systems supporting cross-border payments either already use or have chosen to implement ISO 20022, an open standard for electronic data exchange between financial institutions.[9] Many large financial infrastructures and user groups are already using ISO 20022 in payments and securities settlement, and it has become the standard for real-time payments.[10] Major reserve currency payment systems have adopted or are in the process of adopting the ISO 20022 messaging standard (eg TARGET Services, Fedwire, FedNow, CHAPS and BOJ-NET).

The value of the ISO 20022 messaging standard comes from providing consistency of data in payment messages. Adhering to the data dictionary used to support ISO 20022 messaging will ensure that the data exchanged via APIs are themselves harmonised with the data submitted to payment systems via messaging routes. It will therefore be possible to transfer payments data readily across different messaging channels, without the need for manual intervention and with minimal use of or investment in translation services. The migration of payment systems to a standardised ISO 20022 data model for cross-border payments is a strong enabler for realising the full benefits of API harmonisation.

As the global payments ecosystem moves towards adopting use of APIs and ISO 20022 in messaging, it will be important to emphasise the synergies of coexistence. Given the significant investments made by financial institutions in their existing messaging infrastructure and the expertise developed around messaging standards, harmonised APIs should build upon, leverage and interoperate with these investments. Organisations that have an ISO 20022-driven enterprise data model will have a head start when leveraging those data components to create or support ISO 20022 API-based solutions.

Potential action items:

1. Developers of cross-border payment APIs should familiarise themselves with ISO 20022 in the API context, harmonised ISO 20022 data models and relevant regulatory standards for cross-border transport of payment information, including:
    - ISO 20022 in the API context;[11]
    - The CPMI's ISO 20022 harmonised data model (CPMI (2023));
    - FATF Recommendation 16 on Wire Transfers;

[9] SWIFT oversees the ISO 20022 financial message repository as the formal registration authority on behalf of ISO, publishing and maintaining ISO 20022 message definitions and safeguarding the standard's quality.

[10] The global adoption of ISO 20022 was spurred by SWIFT's decision to adopt it for cross-border payments and to end support for the MT messaging standard in cross-border payments messages exchanged across its private messaging network by November 2025.

[11] For example, the ISO 20022 API resources found on the ISO website: www.iso20022.org/development/status-iso-20022-submissions?block_id=iso20022_submission_catalogue_all_submissions_block&prefix=BJAP.

- CBPR+ and HVPS+ implementation guidelines.[12]

2. API standards organisations and PSPs should include as resources on their developer portals several highly reused ISO 20022 message components that form the foundation of the payment data model and have been identified by the ISO 20022 registration authority. This step should minimise the need for complete overhauls of existing systems and ease coexistence between payment messages and APIs.

3. API standards organisations should cooperate with the ISO 20022 Registration Management Group through its working group for identifying cross-domain best practices to integrate or coexist with other financial messaging standards.

4. PSPs and payment system operators should conduct pilot projects and proof of concept implementations involving both ISO 20022 and harmonised API standards to demonstrate their complementary nature and identify any integration challenges.

5. Industry associations should develop case studies and success stories that showcase how financial institutions have successfully integrated harmonised standards with their existing messaging infrastructure, realising tangible benefits.

2.2.5 Recommendation 5: Security

API developers should incorporate into cross-border payment APIs, especially open APIs, prevalent industry security standards (ie "world-class basics") to ensure that all parties can trust the platform and the payment instructions and to safeguard payments against the risk of misuse and fraud.

Security and compliance are crucial aspects of cross-border payment APIs as they involve sensitive and confidential data (such as personal information, financial details and transaction records) that need to be protected from unauthorised access, misuse, or fraud. Moreover, cross-border payments are subject to various laws and regulations (such as AML, KYC and data protection regulations) that need to be adhered to by the API providers and users. Given its crucial importance, API security should meet the highest standards and API harmonisation efforts should work towards this goal, including by implementing the following API security and compliance measures ("world-class basics"):

i) *Zero-trust architecture:* API developers should implement their APIs with a zero-trust architecture[13] to ensure authentication of the API user and verify a customer's level of access to data. In a zero-trust architecture, access to APIs should follow principles of least privilege by only providing fine-grained access to the required APIs. Industry-standard specifications, like OAuth 2.0 or SAML, can facilitate the implementation of a zero-trust architecture by providing mechanisms for verification, fine-grained access and continuous authentication.

ii) *Logging and audit trails*: API developers should ensure that APIs provide appropriate logging and audit trails to track usage and monitoring. These audit trails can be used to help support API consumers who are facing issues or to investigate any suspicious activity.

iii) *Secure and authenticated communication channels and protocols*: API developers should ensure that they have properly secured their networking and application layers. Technologies, such as Transport Layer Security (TLS) and Hypertext Transfer Protocol Secure (HTTPS), can help secure

[12] SWIFT's MyStandards offers the possibility to export payment resource models in JSON format with ISO 20022 semantics.

[13] A zero-trust architecture is a security framework that assumes no implicit trust in any network or user. It requires strict verification and authentication for every access request, regardless of whether the request originates from within or outside the network perimeter. Zero-trust architecture ensures that API requests are thoroughly authenticated and authorised before granting access to protected resources. In the context of APIs, zero-trust architecture focuses on implementing security controls at multiple levels, such as authentication, authorisation, encryption and monitoring.

communications between clients and the server. A tiered approach to security can help to ensure that the data transmitted is encrypted and secure.

- APIs should also implement multifactor authentication and authorisation mechanisms for access and usage.
- APIs should encrypt and anonymise data that they transmit and store.
- APIs should have an inventory of cryptographic assets so that they can be upgraded and made quantum-safe where necessary.

iv) *Appropriate rate limiting and throttling*: API developers should utilise strategies to mitigate denial of service attacks by applying appropriate rate limiting and throttling to ensure that the service remains operational. These strategies are used to ensure that APIs continue to be available for users even if the service is being attacked.

v) *Updating and patching:* API developers should keep their software up to date to help prevent potential security vulnerabilities from being utilised. Dependency scanning tools can help identify these older packages in their own software. API developers should also use a web application firewall to help with preventing known vulnerabilities, logging and auditing.

vi) *Non-repudiation signatures*: Where conversions between APIs are required to facilitate interoperability for end-to-end payments, consideration should be given to techniques for preserving non-repudiation signatures[14] across API interfaces and to ensuring secure transfer of data across systems in cross-border payments.

Potential action items:

1. API developers should assess API security features against the checklist of "world class basics" in this recommendation.

2.2.6 Recommendation 6: Common registries

API developers should leverage global identification standards for verification and validation of the originator and beneficiary of a transaction to allow for precise, instant and automatic identification across borders, where possible. Among these global identification standards are the business identifier code (BIC) and the legal entity identifier (LEI).

By adopting identifier standards, such as the BIC and the LEI, API developers can enable precise, instant and automatic identification of parties involved in transactions across borders. Allowing multiple or alternative data sources is problematic in today's payments ecosystems, significantly reducing the ability to conduct straight through processing. It is important to emphasise that all identifying data fields, including the legal name, local business registry information, headquarters and legal address can be retrieved from the LEI reference data in an automated manner.

The BIC (ISO 9362) serves as a standardised alphanumeric code used to identify financial institutions worldwide. It facilitates secure communication and routing of financial transactions between banks and other financial institutions. By incorporating BICs into their systems, API developers can promote accurate identification of the financial institutions involved in transactions, enabling seamless interoperability and communication across borders. Similarly, the LEI (ISO 17442) is a globally recognised

[14] The term "non-repudiation signatures" refers to the use of digital signatures to ensure the authenticity and integrity of API requests and responses. They are employed to provide evidence that a specific party initiated an API request and that the data transmitted has not been tampered with during transit. The digital signature provides evidence that the sender cannot deny their involvement in the API request. The private key used for signing is unique to the sender and is forge-proof without access to the private key. Non-repudiation signatures involve several steps, including signature generation, signature verification, data integrity checks and authenticity checks.

identifier used to uniquely identify legal entities participating in financial transactions. LEIs provide a standardised and unambiguous reference for identifying corporations, trusts, government entities and other legal structures involved in financial markets. By supporting the use of LEIs for verifying the identity of legal entities, API developers can enhance transparency, reduce counterparty risk and streamline regulatory reporting in cross-border transactions.

Together, the BIC and LEI can serve as powerful tools for API developers to ensure accurate and reliable identification of transaction counterparties. By incorporating these global identification standards into their systems, API developers can enable seamless and efficient cross-border transactions while enhancing security, transparency and compliance with regulatory requirements.

Potential action items:

1. API developers to seek out resources from global identifier registries on the potential merits of using their identifiers.

2.2.7 Recommendation 7: Developer training and knowledge transfer

Jurisdictional authorities, payment system operators and standards organisations should encourage and facilitate the training of API developers in cross-border payments. The aim should be to increase API developers' knowledge of ISO 20022 and other relevant international payment standards (including regulatory guidance), as well as of cross-border payment processes more generally. Efforts should also be focused on API developers in emerging market and developing economy regions.

Preparing the ground for an ecosystem which supports global cross-border payments presents special challenges to API developers, and success in this area will require coordinated support from payments experts. Developers of APIs for cross-border payments need to consider not only the immediate connection requirements which it is their job to facilitate, but also how the implemented communications protocols are used in cross-border contexts that are yet to be defined and implemented.

These requirements mean that productive API developers in the cross-border space need to find ways of connecting to the global dialogues on cross-border payments. This step is not just a necessity for gathering knowledge, but also includes developing the capacity for joining and shaping the evolving dialogues around cross-border standards and best practices for cross-border payments. These dialogues need the widest participation possible if they are to result in standards which are responsive to customer needs across the whole space of global payments. The standards may at first appear as a fait accompli – to be consumed by API developers. However, contributing to the dialogue from the perspective of their own jurisdictions and use cases is one of the most valuable things that API developers can do. This value is both for the developers' local purposes, where that contribution enables them to set those purposes in the context of a global ecosystem, and for the global payments landscape, which becomes able to evolve to accommodate local requirements that may be invisible in an initial standardisation effort. Efforts should be focused, in particular, on work to accommodate the perspectives of API developers in emerging market and developing economy (EMDE) regions. The challenges and needs of API developers in EMDE regions are likely to be very different from those in more advanced markets, where the technical resources and infrastructures (eg awareness and use of common registries) are more developed.

National and regional jurisdictions, as well as standards organisations, have an important part to play here. They should identify the key areas of standards definition – for instance, the work that SWIFT has done in developing the generalised CBPR+ model for cross-border payments as well as the ISO 20022 standard itself – and make it as easy as possible for API developers to understand and contribute to these initiatives. The standards organisations themselves need to be constantly looking for ways to reach out to and involve a wider community of developers to ensure that their standards reflect the diversity of a payments landscape with customers whose needs differ widely.

Potential action items:

1. PSPs providing cross-border payment API services should devise and implement plans to facilitate greater exchange of information and expertise between business areas and API developers. PSPs should support their API developers to enrol in ISO 20022 training courses and use developer resources provided by standards organisations and industry associations (see related Recommendation 8 and immediate action item on developer resources).
2. International and regional organisations should devise and implement technical assistance modules to promote API harmonisation in cross-border payments and the API harmonisation design principles.
3. Stakeholders should establish working groups or advisory committees that bring together experts from both traditional and harmonised standard domains to facilitate knowledge-sharing and identify potential areas of convergence or conflict.

2.2.8 Recommendation 8: Developer resources

API standards organisations should make available developer-friendly resources, including documentation and tools, to facilitate the development and adoption of harmonised APIs in cross-border payments. These resources should allow developers with non-payments backgrounds to access the necessary information and technical standards for cross-border payment use cases. API developers and consumers should make use of these resources to assist in adoption and to reduce overall errors.

One driver of fragmentation in API standards is inability of API developers to quickly and easily access, understand and use the resources available to them on developer portals of standards organisations or other stakeholders. This can result in API developers inventing their own solutions to fit their particular needs. While such solutions may be convenient over the near term, they lead to undue fragmentation and undermine the longer-term benefits of adopting more harmonised solutions.

Different standards organisations may provide different types of resources on their portals because they may take different approaches to the standardisation of APIs depending on the business context and the needs of the communities they serve. For example, where different implementations need to work identically with common clients – eg in an open banking scenario – API standards typically cover all details of the API, including detailed contact information, security standards to be used, exception handling conventions, etc. In other cases, the aim of the standards organisation is to facilitate data consistency between actors in an end-to-end business process, and technical implementation details are largely out of scope. The standard focuses instead on common business data definitions (semantics) and data types. For example, the ISO 20022 API standardisation initiative aims to produce abstract but detailed API data models for specific business domains and is silent on technical implementation details.

In all cases, standards organisations are recommended to clarify the aims of the standards they develop and promote and the aspects of API implementation that are in scope. Further, they should publish the artifacts of the standard using accessible, well understood formats which are supported by common tools (such as the OAS, YAML and JSON schemas). Specifications should be easy for developers to find, understand and consume, including for those unfamiliar with cross-border payments. Formal material should be supported by informal content to facilitate developers' use of the standard (eg white papers, FAQs, "how-to" guides). Where the aim is to standardise API contracts, developer resources and "sandbox" facilities should be considered. Finally, guidelines should be provided on how a base API can be extended and/or restricted to meet specific market practices while retaining interoperability. API developers should also seek out standards that promise to accelerate development, minimise errors and facilitate interoperability. Where standards fall short of their needs or prove difficult to work with, developers should communicate that feedback to the responsible standards organisations.

Potential action items:

1. API standards organisations should clearly state the intent and scope of the standards they provide, to help developers efficiently identify standards appropriate to their needs.
2. All organisations hosting developer resources on their websites for cross-border payment APIs, especially API standards organisations, should assess whether their websites and resources are developer-friendly, following the checklist provided in the toolkit (see Annex 1).
3. Based on these assessments, all relevant organisations should develop and initiate plans to improve the developer resources on their websites.
4. API standards organisations should encourage developers to provide regular feedback on the developer-friendliness of published resources, and plans to implement improvements where required.

2.2.9 Recommendation 9: Pre-validation

PSPs should prioritise the development and harmonisation of pre-validation APIs that support the validation of key data items within a payment request. API end users should, where available, make use of pre-validation services to reduce the likelihood of failure after initiation.

Pre-validation is designed to verify that details of a cross-border payment are correct before processing begins so that the payment can be processed efficiently without automation breaks. Pre-validation can also ensure that the payer is aware of the terms of the transaction (fees, foreign exchange (FX) rates, predicted time of delivery) before committing to its execution. Standardised APIs are well adapted to the requirements of pre-validation services, which require real-time request/response interactions, and may need to aggregate data from multiple implementations to satisfy a request.

Pre-validation checks include confirmation of payee (COP), ie checking that the payee's account exists and is open to receive funds, how long the payee's account has existed, and also that the name of the account owner matches that of the party the payer wishes to pay. Other elements that can be checked or determined include purpose codes (whether these are required for the destination market, and if so whether an appropriate code has been provided), clearing codes, market infrastructure amount limits and/or operating hours, FX rates and fees, and estimated delivery time.

Potential action items:

1. PSPs should prioritise development of APIs that enable transparency for end users on fees, FX rates and delivery time, and that check that key details of the payment (such as the creditor account) are correct before processing begins.
2. PSPs that offer payment services to end users should take advantage of pre-validation APIs to provide the highest levels of transparency and processing efficiency.
3. Standards organisations should prioritise development of standards for pre-validation APIs, recognising that pre-validation requests may be satisfied by combining data from multiple services and standardisation is key to making this safe and efficient.
4. In jurisdictions where COP services are available for domestic payments, providers of these services should work to extend them to financial institutions and PSPs in originating countries to cater to cross-border payments via pre-validation APIs. These APIs should take into account jurisdictional limitations with respect to payee privacy and security.
5. In jurisdictions where COP services do not exist, authorities, payment system operators and standards organisations should encourage the creation of such a service for domestic and cross-border payments via public-private collaboration. This service should be developed and implemented in an open and neutral manner so as not to disadvantage any stakeholders within a given jurisdiction.

2.2.10 Recommendation 10: Progress tracking and promoting adoption

Jurisdictional authorities should track the progress and facilitate adoption of API harmonisation efforts (including but not limited to the implementation of these recommendations), with the aim to continue promoting harmonisation to enhance cross-border payments.

APIs are a fast-moving technology, where usage is driven by individual needs and use cases that differ across institutions, sectors and jurisdictions. Additionally, maturity with respect to API technological know-how and adoption differs across jurisdictions. These factors mean that harmonisation is not an initiative that can be realised overnight, nor at a consistent speed across jurisdictions. It is likely that harmonisation will progress in accordance with the priorities and capabilities local to the jurisdiction; this should be recognised and managed.

Harmonisation is a longer journey requiring a level of facilitation and involvement from authorities to ensure that improving cross-border payments remains a focus of API harmonisation efforts. Authorities are in a unique position to continue the conversation in promoting the benefits of greater harmonisation and should leverage their influence to achieve this end while taking pains to ensure their role remains facilitative and not directive.

Potential action items:

1. Jurisdictional authorities to distribute the API best practice design principles to payment system stakeholders (eg operators, PSPs, API standards organisations) within their jurisdictions, especially those directly involved in API design.
2. Jurisdictional authorities to consider organising an event in 2025 (eg through the CPMI) with the participation of relevant stakeholders to receive feedback on these recommendations and API best practice design principles.
3. Jurisdictional authorities to consider working with the relevant stakeholders (eg through the CPMI) to form an API steering committee that can oversee the creation of a reference architecture and open source API libraries for financial institutions to communicate with each other using the API best practice design principles. The open source reference libraries can serve as a starting point for financial institutions to further customise the APIs based on the use cases and other considerations.
4. The API steering committee should promote the adoption of API best practice design principles via industry events and establish a governance framework to evolve the API connectivity principles as the industry's needs change.

3. Conclusion

The adoption of APIs in cross-border payments can lead to significant efficiency gains along the payments chain; this includes payment initiation, back office processes, tracking and status, reconciliation and reporting, among other functionalities. However, while APIs hold potential for enhancing cross-border payments, the current fragmentation of API technical standards is substantially limiting this potential. The G20 has identified the harmonisation of APIs for cross-border payments as a priority action. The CPMI convened an expert panel with industry (the API Panel of Experts, or "APEX") to develop the API harmonisation recommendations presented in this report.

The 10 recommendations of this report, grouped into four categories, represent a balance of two key insights from the APEX discussions. First, API harmonisation can benefit from leveraging existing standards and harmonisation initiatives, including by facilitating awareness-raising across expert communities. When an initiative involves bringing together two expert communities with very different skillsets and backgrounds (ie API developers and payment professionals), the marginal gains for API harmonisation from awareness-raising and educational efforts across these two expert communities may

be substantial. Second, there is a natural limit to the achievable scope of API harmonisation at present due to the diversity of underlying use cases, incentives and markets.

The CPMI recognises that the API harmonisation process will be a lengthy one involving a diverse range of stakeholders. As such, it is also recommended that implementation and progress monitoring of suggested action items for each recommendation form an important part of the future CPMI cross-border payments agenda for authorities and private stakeholders.

References

Committee on Payments and Market Infrastructures (CPMI) (2022): *Interlinking payment systems and the role of application programming interfaces: a framework for cross-border payments*, Report to the G20, July.

——— (2023): *Harmonised ISO 20022 data requirements for enhancing cross-border payments*, Report to the G20, October.

Financial Stability Board (FSB) (2020): *Enhancing cross-border payments: stage 3 roadmap,* October.

——— (2023): *G20 roadmap for enhancing cross-border payments: priority actions for achieving the G20 targets,* February.

Fitzgerald E, A Iles, T Lammer (2024): "Steady as we go: results of the 2023 CPMI cross-border payments monitoring survey", *CPMI Briefs*, June.

Annex 1: Toolkit

A1.1 Best practice design principles for APIs

A1.1.1 Data principles

Principle 1: Reuse dominant existing international standards for reference data

Wherever possible, API designers should use international standards which are universal or predominant for the encoded content of reference data. The use of reference data which is newly generated ad hoc should be a last resort. Some of the existing, well defined international standards include: standardised codes for countries (ISO 3166 Country codes), currencies (ISO 4217 Currency codes), party identifiers (ISO 17442 Legal entity identifier) and market infrastructures (ISO 10383, Market identifier code). These standards are implemented in industry back office systems and business processes. Reusing these standards will make APIs easier to implement and simplify end-to-end interoperability. Note that some markets do *not* use international standards, so API designs may need to accommodate local alternatives (see Principles 6, 7).

Principle 2: Reuse dominant existing international standards for data semantics

The industry has adopted standardised business data semantics for cross-border payments (eg ISO 20022 for correspondent banking and ISO 8583 for credit cards). APIs should reuse standard business definitions and data types to ensure end-to-end interoperability and compatibility with existing infrastructure. API designers and implementers should provide publicly available documentation mapping their representations onto general business semantics.

Principle 3: Reuse standard resource models where available

ISO 20022 includes a number of standardised API resource models in the payments domain, and more are under development. For API designs that are intended for broad usage, designers should consider using, and where necessary extending, these models. Implementers can liaise with the submitter of the standard resource model and the ISO 20022 registration authority to propose changes or new resources for future releases. Reusing ISO 20022 resource models ensures easy compliance with Principles 1 and 2.

Principle 4: Build in upfront checks and pre-validation

Wherever possible, cross-border transactions should be (pre-)validated to maximise the certainty that they will complete successfully before the parties commit to execution. It should always be possible to verify that a payment can be executed before any party is required to commit funds. Pre-validation APIs should include:

- Metadata APIs – mandatory fields, conditional fields and value lists;
- Verification of payee (where applicable);
- Payment payload (ie other data sent with a payment request).

Principle 5: Exchange only relevant data

Pre-validation checks should ensure that only data relevant to the market in which the transaction is to be settled is included in the transaction, and that market-specific elements (eg purpose codes, balance of payment codes, etc) are compatible with the target market.

Principle 6: Design for extensibility

APIs should be designed for extensibility (ie extending an API specification to be able to include additional information that may not be fully defined in the specification, and not necessarily technology-specific extensibility). In particular, it should be possible for communities of users to add content and capabilities without breaking the implementations of other users.

Principle 7: Consider data privacy

API designs and implementations need to consider data privacy concerns in the markets in which they are to be used. Concerns include compliance with regulations on data protection, data residency regulations, payment system regulations, etc. Given the global nature of the cross-border payments business, API designers should be conservative in their assumptions of what data – particularly personal data – can be captured, stored and processed, and should seek legal advice from experts in the field.

API developers should consider incorporating a "call-back function" into API designs. A call-back function would allow parties to the payment to request the information they require, rather than transporting the aggregate of all information required as part of the message. This structure should also support zero-knowledge proof (ZKP) techniques[15] to reduce the need for parties to see the information.

A1.1.2 Business principles

Principle 8: Consider the end-to-end context

Developers of API standards should consider the end-to-end flow of data and control and ensure their API fits into the bigger process. This is because APIs for cross-border payments will in most cases either introduce new steps or supplant existing steps in a multi-step, end-to-end business process involving other parties and markets using their own APIs. Thus, end-to-end cross-border transactions may require use of messages and APIs from different implementers. Alignment on business codes and semantics (Principles 1, 2) is one way to facilitate end-to-end interoperability. Where conversions between APIs are required to facilitate interoperability for end-to-end payments, consideration should also be given to techniques for preserving non-repudiation signatures across API interfaces.

Principle 9: Have a plan for adoption

Any initiative to introduce standardised APIs for cross-border payments must consider carefully the mechanics of rollout and adoption if it is to succeed. Many models are possible. In some cases, a single authority can impose a solution on participants (eg a closed loop system). In others, adoption is voluntary, and participants need to be persuaded to implement, for example by the commercial opportunity the service represents, competitive pressure, etc. Anyone proposing a new API or API standard should be able to describe how it will be adopted in the market.

[15] ZKP is a cryptographic protocol that allows one party (the prover) to prove to another party (the verifier) that a statement is true, without revealing any additional information beyond the truth of the statement itself. ZKPs are designed to ensure privacy and confidentiality while still providing trust and verification. They are commonly used in various applications such as authentication, identity management, secure communication, and blockchain technologies.

Principle 10: Drive for harmonisation

Differences between markets and market-specific practices will always exist, but in many cases, apparent differences may prove to be superficial. Designers of open APIs that are expected to be implemented by multiple actors should seek to identify superficial differences and impose a harmonised implementation where possible.

Principle 11: Accommodate market variation in a structured and predictable way

Where more fundamental differences exist (eg exchange controls in certain markets, differences in clearing arrangements, local character sets), API design should ensure that deviations from a shared core data set and/or behaviour are isolated and easy to identify as such.

Principle 12: Consider FX conversion

Many cross-border transactions will include a currency conversion step. One of the techniques for reducing the overall cost of cross-border payments is to ensure that currency conversion can be obtained on the terms most advantageous to the end users. This implies that currency conversion can happen at one of several points in the process – at the debtor agent, intermediary or creditor agent, or via a third-party FX provider. API implementations should be flexible to accept different models of FX conversion, and support mechanisms to exchange information on FX.

A1.1.3 Technical principles

Principle 13: Follow established industry design patterns and conventions

APIs are more likely to be implemented, and implemented correctly, if their design is consistent with established industry design patterns and conventions. For example, API designs should be based on established data models, interaction patterns, style guides and naming conventions. Moreover, all singular API interactions should be designed to be stateless;[16] that is, API calls should not rely on previous interactions with a server to receive an appropriate response. This design approach offers many advantages: since the server never stores the state, it can be easily scaled up to handle an influx of users, and can offer improved resiliency for the system.

Principle 14: Make APIs "developer-friendly"

APIs should be developer-friendly to allow easy onboarding. Providing documentation and tools will make it easy for both developers and business individuals to proceed. APIs should follow easy-to-consume specifications expressed using industry standards. These specifications can be defined in a variety of tools, eg OpenAPI specifications, and can be used to define APIs in both human and machine-readable fashion.

Principle 15: Align with prevalent API security standards and best practice

APIs need to align to prevalent API security industry standards. They should be highly secure, require multiple vectors of protection and follow the principle of least privilege access to data.

Principle 16: Define service-level agreement (SLA) thresholds

Specifications should include SLA thresholds, eg for response times.

[16] Stateless APIs are APIs that do not store any session or "state" information (ie data that is maintained by the server during the course of a client's interaction with an API) between requests. In a stateless API, each request is independent and self-contained, containing all the necessary information for the server to process it without relying on any previous requests or stored session data.

Principle 17: Design APIs to be implementation-agnostic

APIs should be designed to be agnostic to any technical or vendor implementation.

Principle 18: Provide robust version control

APIs need to provide a clear and consistent governance policy around versions and their life cycle. These policies should consider the following:

- APIs should be able to evolve but at the same time should strive to maintain backwards compatibility. The expectation will be that consumers will use the APIs in a forward-compatible way. For example, an optional data element can be added to an API and this can be implemented and used by existing consumers without any change.
- Breaking changes will on occasion be inevitable. A breaking change would introduce another technical endpoint, leaving the older version also operating in parallel (for a time). In this case, API consumers (ie clients or applications that interact with the API to access its services or data) should be protected from being affected when these changes are initially rolled out.
- APIs should be life cycle managed: a clear policy should be set on life cycle management, deprecation of prior versions and obligations on consumers. Tooling (ideally the API developer portal) should be available to help consumers manage and gain visibility into API life cycles.

A1.2 Examples of APIs following and not following the API best practice design principles

This section provides practical examples for API developers of code that follows or does not follow the API best practice design principles in Annex A1.1. These examples are for illustrative purposes only.

Example #1: API not following best practice design principles

The following documentation is provided to API consumers as a way to submit payments:

API endpoint: GET /v1/payments
This API is used to submit a payment.
Request body:


```
{
   "dbt": {
       "name": "John Doe",
       "favourite_color": "blue",
       "occupation": "developer",
       "account": "1234",
       "financial_institution": "AAAACA00"
   },
   "crd": {
       "name": "ACME Corporation",
       "account": "9876",
       "financial_institution": "SAMPLE_LEI"
   },
   "amount": {
       "currency": "CAN",
       "value": "1000"
   },
   "auth": {
       "key": "8f80c0ec-a7b5-4230-b344-99aeab4b09bf",
       "secret": "f61cef71-75ee-462f-902e-fc0b70c63ab1",
       "role": "ADMIN",
       "financial_institution": "Sample Bank"
   }
}
```


Payment object:

dbt	The debtor.

crd	The creditor.
amount	Object representing the amount.
auth	Used to validate the caller and their role.

Amount object:

currency	Represents the currency.
value	How much the debtor is sending to the creditor.

Responses:
Success – 200:

```
{
   "transaction_id": "5f6657e7-b860-4e23-885e-7a550daf911a",
   "status": "ACCP",
   "amount": {
       "currency": "CAD",
       "value": "10.00"
   },
   "debtor": {
       "name": "John Doe"
   },
   "creditor": {
       "name": "ACME Corporation"
   }
}
```


Transaction object:
[excluding transaction definitions for brevity]

Taking a look at this API, the following issues can be identified:

- Inconsistent naming: In the request, the caller must send the debtor field as "dbt", but then receives the debtor field labelled as "debtor".
- Unnecessary personal information is passed on, like the debtor's occupation and favourite colour.
- Inconsistent use of fields: The amount's value has no decimals in the request, but in the response is returned with a decimal.
- Security processes for authentication are not followed correctly: a role and a financial institution are passed on, but it is not clear why. This can lead to an attacker trying to use those fields on behalf of other people. An API consumer may also recognise the potential flaw and choose not to interact with that system. In addition, passing on a set of static credentials is generally not

viewed as good practice as those values can be accidentally leaked and then used by a different actor at a later time.

- Improper use of RESTful verbs: an HTTP GET should not be used to submit or cause side effects to a system. As well, not all systems support GETs with a body.
- The "financial_institution" field in the creditor and debtor messages contain different data; one uses a BIC and the other uses an LEI. It is not clear to an API consumer if that is always the case, or if the system attempts to be "smart" about reading the field.
- Consider modifying this API with the following:
- Remove the authentication field. Consider moving the authentication to its own authentication server that provides an expiring token like a JWT, which can be verified for authenticity by the API and can no longer be used if expired.
- Standardise naming, not only in this API but in all the APIs in the catalogue.
- Remove unnecessary personal information.
- Change the HTTP verb to POST.

Example #2: API following best practice design principles

The following documentation is provided to API consumers as a way to get details of a transaction:

API endpoint: GET /payments/{transaction_id}
Parameters:

transaction_id	The unique end-to-end transaction reference (UETR) for the transaction.

Responses:
Success – 200:


```
{
   "transaction_id": "5f6657e7-b860-4e23-885e-7a550daf911a",
   "status": "ACCP",
   "amount": {
       "currency": "CAD",
       "value": "10.00"
   },
   "debtor": {
       "name": "John Doe",
       "account": "1234",
       "financial_institution": "AAAACA00"
   },
   "creditor": {
       "name": "Jane Doe",
       "account": "9876",
```

```
        "financial_institution": "BBBBGB11"
    }
}
```

Transaction object:

transaction_id	The UETR for the transaction.
status	The status code for the transaction; the code represents an ISO 20022 external code set.
amount	An object representing the amount and currency that has been sent.
debtor	A party object that represents the party that owes money.
creditor	A party object that represents the party to whom the money is owed.

Amount object:

currency	An ISO 4217 code representing a currency.
amount	A string representing the amount the debtor is sending; this amount will be positive and have at most two digits.

Party object:

name	The party's name.
account	The party's account number.
financial_institution	The BIC for the debtor/creditor.

The following API does quite a few things well:

- It is easy to read and understand.
- All the fields in the response as well as expected fields have a clear and understandable definition.
- It uses proper RESTful techniques to retrieve data.
- It uses consistent naming. For example, "transaction_ID" is consistently named in both the parameters and the response.
- It uses standards to define the status, currency and financial institution.

A few key details are missing though:

- What does this API do? Although this may be obvious to the API developer, not every API consumer may have full understanding of what it does.
- Is this API versioned? If so, how does an API consumer pass it in? And what is the policy?
- What kind of errors could be returned? If so, what do they look like?

Thus, this API may be enhanced by adding the following information to its documentation:

API endpoint: GET /v1/payments/{transaction_id}
This API is used to retrieve details on a payment. It provides details on the debtor and creditor as well as the payment status.

> Note: This API has been deprecated as of 2024-06-07, and is scheduled for removal one year from that date. Please migrate to a newer version of the API. For further information on our versioning and deprecation policy, please visit the Versioning page.

Parameters: [excluded for brevity]
Responses:
Success – 200: [excluded for brevity]
Failures:
404 – Not found:

```
{
   "code": "NOT_FOUND",
   "message": "Transaction was not found or you do not have permissions for the transaction"
}
```


429 - Rate limited:

```
{
   "code": "RATE_LIMITED",
   "message": "This request has been rate limited. The number of calls has exceeded the threshold of 60 per minute."
}
```


[excluding additional errors for brevity]

Error object:

code	An error code providing a short description of the error. A full list of error codes can be found on the error codes page.
message	The full error message providing additional details as to why the API call failed.

A1.3 Checklist for developer-friendly portals

A developer portal – often shortened to "DevPortal" – is the interface between a set of APIs, SDKs or other interactive digital tools and their various stakeholders. A developer portal provides (internal and external) developers with all the necessary information and tools to integrate with an API. It is a central hub for developers to access helpful guides, use cases and other resources to help them understand and "consume" the API. A user-friendly developer portal can make a significant difference in attracting and retaining developers.

In discussions with developers with extensive experience in open banking, the main feedback received was that what makes a developer portal "good" is the availability of a tutorial/clear documentation on onboarding, the authorisation and authentication flows, the API endpoints and the URL of the sandbox. Basically, these developers want to get started with the lowest barrier possible, so the amount of self-service a portal offers is definitely a success factor. Developers also expressed preference for quick messaging response times to their technical questions as opposed to responses that take hours or even days.

This document describes some of the elements which make a developer portal provide a positive developer experience.

Best practices

- A developer portal is more than just a list of endpoints

A developer portal needs to explain the business value of the APIs and the business processes and use cases in which these APIs are used on top of the technical documentation.

- A developer portal is self-service by design

A developer should be able to access the documentation easily and start with design and development based on the documentation and tools provided. Elements to consider are the automation of provisioning of API keys and tutorials to help developers with onboarding, as well as a sandbox environment where they can test their integrations.

- Active management of the community

A developer portal is one method of getting developers to engage with APIs. Offering support to the developer community and engaging with them to get feedback and enhance the products and APIs is essential. Tools such as a Slack channel, whereby developers can interact, can be very valuable, as can implementing notifications that alert developers to upcoming changes.

- Non-functionals

Ensure your developer portal is secure and performs well. Best practices include using secure methods for API authentication and authorisation, such as OAuth 2.0, implementing rate limiting to prevent abuse, and designing the API infrastructure to be resilient, with failover mechanisms to ensure high availability.

- Ease of use

Finally, make sure the developer portal is easy to use. This includes making the documentation easy to find, ensuring the developer portal works on different screen sizes and different devices (including tablets and mobile phones), has a search function and tutorials. The main error codes and failure scenarios should be included, as should possible solutions, as well as metrics that help developers understand things like volumes, errors and performance. Provide SDKs and libraries in popular programming languages to help developers integrate with your APIs more easily.

Checklist

The following checklist can be used to validate the quality of the developer portal:

- Completeness
  - Are there real-world use cases and code samples in multiple programming languages?
  - Is the API reference documentation detailed and complete, including endpoints, parameters, request/response formats, and examples?
- Self-service
  - Is a sandbox environment available for safe testing without affecting production data?
  - Is provisioning of API keys automated?
  - Are there clear and comprehensive guides to help new users start quickly?
- Developer engagement
  - Is there an active community or forum for peer support?
  - Are multiple support channels (Slack, email, chat, ticketing) available?
  - Are there ways for developers to provide feedback on the APIs and the portal?
- Non-functionals
  - Are secure methods like OAuth 2.0 or API keys used for authentication and authorisation?
  - Rate limiting and throttling: Are rate limiting and throttling clearly documented and enforced?
  - Is there redundancy and failover infrastructure to ensure high availability?
- Ease of use
  - Does the portal work well on different devices and screen sizes?
  - Is there a robust search feature that helps users find information quickly?
  - Is there a comprehensive FAQ section and troubleshooting guides?
  - Is there thorough documentation on error codes and troubleshooting steps?
  - Are usage analytics provided to give insights into API performance and usage?
  - Are SDKs and libraries available in popular programming languages?

A1.4 Checklist for open and robust governance of API standards

API standards organisations may wish to review recommended guidance with open governance principles.

- *Inclusivity of stakeholders:* The principles and practices that guide the decision-making processes should allow for balanced and fair market representation from multiple stakeholders from the supply and demand sides of the market, allowing them to participate in the governance and decision-making processes (without exclusive domination or guidance), ensuring accountability and promoting transparency. This inclusivity ensures that diverse perspectives are considered and that the governance framework reflects the interests of all stakeholders.
- *Transparency, disclosure and non-discrimination:* Open governance also requires transparency in operations, policies and decision-making processes. This includes making information about governance structures, rules and procedures publicly available. Transparent disclosure of API deliverables (available for free) is also essential to build trust among stakeholders. It also requires non-discriminatory and objective admission criteria.
- *Clear scope definition:* It is suggested that governance focus on technical and organisational requirements, independent from scheme and business arrangements.
- *Clear rules and standards:* Open governance entails the establishment of clear rules, standards and guidelines for open banking participants. This helps ensure consistency, interoperability and fairness in the ecosystem. Participation rules and actual participation should be transparent to the external world.
- *Collaboration and co-creation:* Open governance encourages collaboration and co-creation among stakeholders. It fosters partnerships and cooperation between traditional banks, fintech firms and other relevant parties. This collaborative approach can lead to innovative solutions, improved customer experiences and enhanced competition.
- *Industry standards:* Ensure that the API standards adhere to widely accepted industry standards. This should be embedded in the overall governance.
- *Security and privacy:* Guarantee the integration of robust security measures and privacy controls within the API standards to safeguard user data and transactions, as a fundamental principle in the overarching governance structure.
- *Governance framework:* Ensure that the API standards offer guidance for governance, encompassing roles and responsibilities, conflict resolution procedures and compliance surveillance.

Annex 2: API standardisation initiatives around the world

The growing adoption of APIs in the financial services industry has prompted a variety of standardisation initiatives around the world, often in the context of open banking initiatives. Some prominent examples are summarised below:

- **Australia:** The first phase of the Open Banking Reform in Australia was introduced in July 2019 with customers of the big four banks empowered to provide access to their transaction, deposit, and debit and credit card data to accredited financial services providers. Subsequently, the product scope was enlarged in two additional phases allowing customers to share account and transaction data for home loans or cash management accounts. In July 2021, all other Australian banks provided account and transaction data on phase 1 products. The Australian Data Standards Body (DSB) designed the relevant standards as dedicated API standards that govern how APIs must be built and cover details such as errors and payload structures.
- **European Union and EEA (European Economic Area):** From September 2019, over 5,000 European banks were expected to provide access to account information and payment initiation via open APIs according to the second Payment Services Directive (PSD2). Several European working groups have been developing API standards, the most dominant of which is the Berlin Group, which covers over 75% of all European banks. The primary focus of the Berlin Group standardisation is the Single Euro Payments Area (SEPA). The Berlin Group is open for participation to any market supply side bank, banking association, payment association, payment scheme and interbank processor active in the SEPA payment industry.
- **Hong Kong SAR:** The HKMA published its Open API Framework for the Hong Kong Banking Sector in July 2018 after a public consultation. The Framework takes a risk-based principle and four-phase approach to implementing various open API functions and recommends prevailing industry technical and security standards to ensure fast and safe adoption. It also lays out detailed expectations on how banks should onboard and maintain relationships with third-party service providers in a manner that ensures consumer protection. The Framework serves as an important guide for the banking industry in Hong Kong to adopt APIs effectively.
- **India:** India has kickstarted its approach to open banking by empowering intermediaries to be responsible for customers' consent management. These intermediaries are licensed as non-banking financial companies. In September 2016, the Reserve Bank of India announced the creation of a new licensed entity called Account Aggregator (AA) and allowed this entity to consolidate the financial information of a customer held with different financial entities, spread across financial sector regulators. In India, AA acts as an intermediary between financial information providers (FIPs) such as banks and other holders of customer financial information, and financial information users (FIUs), which are entities registered with and regulated by any financial sector regulator. The flow of information takes place through appropriate APIs.
- **United Kingdom:** The Competition and Markets Authority (CMA) mandated the United Kingdom's nine largest banks to provide standardised open APIs from January 2018. The Open Banking Implementation Entity (OBIE) defined the APIs within the UK Open Banking Standard. This standard also covers areas which are not regulated by the PSD2 (eg credit data).
- **United States:** In 2017, the National Automated Clearinghouse Association (NACHA), in collaboration with the API Standardization Industry Group (ASIG), announced the development of several standardised API use cases. In July 2018, together with the IFX Forum (now Afinis), the ASIG published an implementation concept for RESTful APIs. Afinis is a membership-based governance organisation that brings together diverse collaborators to develop implementable, interoperable and portable financial services standards. Available standardised APIs cover, for example, account validation, payment initiation, account balances and transaction history as well as transaction status.

Annex 3: Composition of CPMI Messaging Workstream and API Panel of Experts (APEX)

CPMI Messaging Workstream

Chair

Ellis Connolly (Reserve Bank of Australia)

Members

Reserve Bank of Australia	Michael Shen* Warren Wise
Central Bank of Brazil	Fernando Machado Cavlcanti
Bank of Canada	Banquo Yuen Seyed Bostandoustnik
People's Bank of China	Jing Huang Zeyang Yu Xue Chen (from May 2023)
Deutsche Bundesbank	Steffen Fährmann* Dirk Beiermann
Bank of England	Helen Bygrave Mark Streather*
European Central Bank	Marek Kozok* Jean Clement
Hong Kong Monetary Authority	Kitty Lai George Chou
Reserve Bank of India	R Karpagam
Bank Indonesia	Jultarda Hutagalung Novyanto Elyana K Widyasari Franz Hansa
Bank of Italy	Riccardo Mancini Carlo Cafarotti
Bank of Japan	Tomohiro Sugo Hiroyuki Nishizawa Takehisa Kanaguchi Maya Okamoto
Bank of Korea	Jisoon Park Seungmin Lee Minji Go

Bank of Mexico	Daniel Garrido Delgadillo
South African Reserve Bank	Magedi-Titus Thokwane Hein Timoti
Central Bank of the Republic of Türkiye	Burçin Bostan Körpeoğlu Serkan Tekbacak
Board of Governors of the Federal Reserve System	Michael Derry* Amber Seira*
Federal Reserve Bank of New York	Frank Van Driessche

Observers

Basel Committee on Banking Supervision	Stefan Hohl
Financial Stability Board	Kris Natoli (until July 2024)
Financial Action Task Force	Ken Menz
International Monetary Fund	Agnija Jekabsone
World Bank Group	Srinivas Gynedi

Secretariat

CPMI Secretariat	Mark Choi

*Member of CPMI API Panel of Experts (APEX).

CPMI API Panel of Experts (APEX)

Chair

William Lovell (Bank of England)

Members from central banks

Michael Shen (Reserve Bank of Australia)

Marek Kozok and Jean Clement (European Central Bank)

Julia Gabriel and Steffen Fährmann (Deutsche Bundesbank)

Zoran Georgiev (National Bank of the Republic of North Macedonia)

Ajab Ali (State Bank of Pakistan)

Michael Derry and Amber Seira (Board of Governors of the Federal Reserve System)

Members from industry

Andrew Saunders (National Australia Bank)

May Lam (Emerging Payments Association Asia)

Stephen Lindsay (SWIFT)

Hussain Farbotko-Merabaksh (Payments Canada)

Christoph Schneider (GLEIF)

Ortwin Scheja and Wijnand Machielse (Berlin Group)

Ferko Weerman (Deutsche Bank)

Eran Vitkon (Finastra)

Vinod John (National Payments Corporation of India)

Rick DuVall (until March 2024) and Kris Ketels (ISO 20022 RMG API SEG)

Martin Walder (SIX BBS)

Michael Richards (Mojaloop)

Nilesh Dusane (Amazon Web Services)

Secretariat

Mark Choi (CPMI Secretariat)

Mark Streather (Bank of England)

Glossary

The report uses the following definitions:

Application programming interfaces (APIs) – a set of rules and specifications for software programs to communicate with each other, that forms an interface between different programs to facilitate their interaction.

- **Open API** – an interface that provides a means of accessing data based on a public standard. Also known as external API.

- **Private API** – an interface that provides a means of accessing data based on a private standard. Also known as internal/closed API.

Authentication – authentication is the act of validating that users are whom they claim to be.

Customer-permissioned data – retail customer data held by banks (eg customer transactions, personal identification data and customer financial history) that is permissioned by the bank's customer to be accessed by a third party (and possibly shared onwards with fourth parties).

HTML – HTML (Hypertext Markup Language) is a formatting system for displaying material retrieved over the Internet.

HTTP – HTTP (Hypertext Transfer Protocol) is a method for encoding and transporting information between a client (eg a web browser) and a web server. HTTP is the primary protocol for transmission of information across the internet.

HTTPS – HTTPS (Hypertext Transfer Protocol Secure) is the secure version of HTTP that uses the SSL/TLS protocol to encrypt data sent between a client and a web server.

JSON – JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax.

JWT – JWT (JSON Web Token) is an open standard used to share information between two parties securely (eg between a client and a server).

OAuth – OAuth is an open-standard authorisation protocol or framework that provides applications the ability for "secure designated access".

Open banking – the sharing and leveraging of customer-permissioned data by banks with third-party developers and firms to build applications and services, such as those that provide real-time payments, greater financial transparency options for account holders, and market and cross-selling opportunities. Individual jurisdictions may define open banking differently.

RAML – RESTful API Modeling Language (RAML) is a YAML-based modelling language to describe RESTful APIs.

REST – a REST (representational state transfer) API is an API that conforms to the design principles of the REST architectural style.

Security – security for information technology (IT) refers to the methods, tools and personnel used to defend an organisation's digital assets.

SOAP – SOAP (Simple Object Access Protocol) is a messaging standard defined by the World Wide Web Consortium and its member editors. SOAP uses an XML data format to declare its request and response messages, relying on XML Schema and other technologies to enforce the structure of its payloads.

Standardisation – standardisation is the process of developing, promoting and possibly mandating standards-based and compatible technologies and processes within a given industry.

Third party – any external legal entity that is not a part of the supervised banking organisation. Third parties can be supervised entities (eg banks, other regulated financial firms) or non-supervised entities (eg financial technology firms, data aggregators, commercial partners, vendors, other non-financial payment firms).

TLS – Transport Layer Security (TLS) is a widely adopted security protocol designed to facilitate privacy and data security for communications over the internet.

UETR – UETR (Unique End-to-end Transaction Reference) is a string of 36 unique characters attached to a payment message that serves as a unique tracking number for a payment as it moves from initiation to final credit.

Versioning – versioning is the creation and management of multiple releases of a product, all of which have the same general function but the later releases of which are improved, upgraded or customised.

XML – XML (Extensible Markup Language) is a markup language similar to HTML, but without predefined tags to use.

YAML – YAML (YAML Ain't Markup Language) is a data-oriented language structure used as the input format for diverse software applications.

Bank for International Settlements (BIS)

ISBN (online) 978-92-9259-781-8