December 2020

White Paper: Considerations for Mandating Open Interfaces

Internet Society

Acknowledgements

The report was developed by a small team led by Carl Gahnberg, following the strategic guidance of Robin Wilton, Olaf Kolkman and Steve Olshansky. Simon Phipps and Javier Ruiz, consultants for the Internet Society, served as lead authors of the report, providing invaluable expertise and vision. We would also like to thank the group of external experts that provided important comments and suggestions in reviews of earlier drafts, including: Amelia Andersdotter; John Bergmayer; Ian Brown; Cory Doctorow; Gurshabad Grover; Ali Lange; Mark Nottingham; Thomas Reibe; and Chris Riley.

Special thanks to Andrew Sullivan, Joseph Lorenzo Hall, and Katie Watson Jordan for review and comments that helped improve the final version of the report.

Table of contents

Introduction 4

Background 6

Motivations for mandating open interfaces 6

Enhancing consumer rights and choice 6

Countering concentration due to network effects 7

Addressing bottlenecks for innovation 7

Key concepts and technical fundamentals 8

Defining "interface", "interoperability" and "implementation" 8

The mechanics of software: data models and state machines 9

The interface in practice 10

Practical considerations 13

Technical aspects 13

Intended outcome 13

Locus of control 15

Developing the interface 18

Shared interfaces and approaches to standards 18

Shared implementations and open source 19

Granting rights—frand and rights waivers 19

Operating the interface 20

Managing change and use 20

Managing access 21

Policy considerations 23

The impact on market dynamics 23

From policy to practice 24

Conflicting objectives and legal clashes 26

Safeguarding security and privacy 29

Summary 31

Appendix A—instant messaging's "family tree" 32

Introduction

As the Internet has become more and more crucial to daily life, businesses, and national economies, there has been increasing discussion about mandating "open interfaces" or "interoperability" among software services and systems[1] and, more recently, even with regards to applications referred to as Artificial Intelligence (AI).[2]

The motivation for these mandates is to prevent service providers from eliminating user choice, therefore avoiding what are sometimes described as "effective monopolies". However, this kind of intervention may be considered even if the situation doesn't trigger antitrust law. In some cases, the desire for mandates is also due to secondary effects—for instance, as society's reliance on digital connections and data grows, so does the political motivation to mandate open interfaces to services and data sources.[3]

This document intends to support policy practitioners involved in, or with the responsibility for, developing legislation, regulation or other public policy measures (education, training etc.), in technology innovation. It explores the technology and policy issues, with numerous references for readers wanting to learn more.

Mandating open interfaces is significant because:

- If done well, it delivers economic, social and technical benefit, reduces the risk of market failure and stimulates sustainable innovation;
- If done badly, it threatens the outcomes listed above and risks creating unintended consequences that put other policy goals in jeopardy.

The goal of this paper is to inform the debate, and to minimize the risk of bad outcomes from any policy initiatives relating to open interfaces. To do this, we will outline the context in which such initiatives are happening, and then describe the practical and policy-related considerations we believe they raise. We conclude with a set of key takeaways to support future questions and concerns.

Open interfaces and the Internet—why it matters

The Internet owes its success not only to a set of technologies, but also to the way in which it operates and evolves. It is fundamentally a model for how to interconnect independent networks into a greater whole, which we refer to as the "Internet Way of Networking".[4]

A critical property of this networking model is the existence of "an open architecture of interoperable and reusable building blocks". These are technical specifications and systems that each deliver a specific function or service to support the interconnection of the networks and the delivery of Internet applications. For instance, an application designer doesn't have to start

1 Crémer, de Montjoye & Schweitzer (2019), "Competition Policy for the digital era", European Commission, https://ec.europa.eu/competition/publications/reports/kd0419345enn.pdf

2 Marsden, C & Nicholls, R (2019), "Interoperability: A solution to regulating AI and social media platforms", Computers & Law October 2019, https://www.scl.org/articles/10662-interoperability-a-solution-to-regulating-ai-and-social-media-platforms

3 Ibid.

4 Internet Society (2020), "The Internet Way of Networking: Defining the critical properties of the Internet", https://www.internetsociety.org/resources/doc/2020/internet-impact-assessment-toolkit/critical-properties-of-the-internet

from first principles and wonder about transmission or routing in the underlying network but can rely on foundational protocols like TCP/IP to facilitate end-to-end communication between the server and a client. Similarly, the TLS protocol provides a defined security service to any application, eliminating the need to invent this mechanism from scratch.

The technical interfaces that are the subject of this paper are, in a similar way, building blocks for developing new technologies and innovations. They can support the Internet's "generative" nature[5] by allowing the application designer to use functionalities and services without details of the underlying mechanics, and to build on existing solutions. Yet, they are also different.

The mandated interfaces we discuss in this paper relate to ways to make proprietary services interact with other Internet applications. Some may argue that such measures don't affect the Internet architecture itself, but merely govern relations among applications that use the Internet. However, there is a risk that such a view is too narrow, because as one Internet application becomes dependent on the service delivered by another application's interface, that interface becomes practically indistinguishable from the other building blocks needed to deliver applications. Once access to an interface is openly provided, new uses—especially in combination with additional sources of data or other resources—may unexpectedly become important. What was merely a convenient route to shared data and events may become a practically indispensable part of the application 'infrastructure'.

In this sense, mandating an open interface should be unequivocally positive, because it aims to produce openness and interoperable systems and safeguard users from potential abuse by dominant actors. However, the open architecture of the Internet means that interoperable building blocks are developed and deployed through mutual agreement and voluntary adoption. And while such a norm should not be seen as a rejection of mandates, a number of practical and policy-related issues can arise when technical building blocks are not based on voluntary or shared incentives—issues that may affect the expected outcome.

5 Zittrain, J. (2006), "The Generative Internet", Harvard Law Review, https://dash.harvard.edu/handle/1/9385626

Background

The case for mandating open interfaces is grounded in increasing concerns about consolidation and monopolies in the Internet economy. This section gives an overview of the main motivations and thinking behind current proposals and serves as a background note to contextualize the paper as a whole. It also introduces key concepts used in the later analysis of the practical considerations for mandating open interfaces.

Motivations for mandating open interfaces

Policymakers, academics and other thought leaders across the world have called for software services and systems to be legally required to provide "open interfaces" and "interoperability" to solve the problem of concentrated market power among a small number of technology companies. Below we summarize some of the main arguments used to support the case:

Enhancing consumer rights and choice

In a well-functioning market, consumers can switch between service providers with little or no friction, particularly if they are dissatisfied with their existing one. However, when switching is too costly or difficult and they are locked into a service provider, this reduces or even eliminates the market incentive to offer competitive choice and value.

Data portability, which aims to simplify consumer choice and ease switching between services, is an example of efforts to mitigate this problem. It is included in existing laws including Australia's Consumer Data Right (CDR)[6], the European Union's General Data Protection Regulation (GDPR)[7] and the California Consumer Privacy Act[8]—however, it remains unclear whether data portability laws alone can change consumer behaviour.

For instance, existing provisions in the GDPR only specify that an individual has the right to receive their data in "a structured, commonly used and machine-readable format" and, "where technically feasible", to have this data transmitted directly to another service.[9] As a consequence, data portability often results in one-off exports of data that the user then needs to manage themselves, which makes switching cumbersome. Furthermore, these provisions provide no guarantees that the recipient service can use the data to deliver a better user experience.

Some of the main technology companies, including Facebook and Google, are developing a joint framework—the Data Transfer Project[10]—to simplify sending user data, such as photos,

6 Treasury Laws Amendment (Consumer Data Right) Bill 2019, https://www.legislation.gov.au/Details/C2019B00025

7 Article 20, EU General Data Protection Regulation (EU-GDPR), https://eur-lex.europa.eu/eli/reg/2016/679/oj For more background on the current issues, see the project between UCL and Open Rights Group, https://blogs.ucl.ac.uk/steapp/2019/11/25/data-portability

8 California Consumer Privacy Act of 2018 (CCPA), https://oag.ca.gov/privacy/ccpa

9 Article 20, EU General Data Protection Regulation (EU-GDPR), https://eur-lex.europa.eu/eli/reg/2016/679/oj

10 Data Transfer Project (DTP), https://datatransferproject.dev

across systems. This is an open source initiative that welcomes all interested parties to join the project, but it remains to be seen if such voluntary mechanisms will lead to improved consumer choice, or whether they prove to be "necessary but insufficient".

Countering concentration due to network effects

The presence of network effects in many online services, where the value of the service increases as the number of users increases[11], is a driving force of concentration in digital markets. Network effects discourage users from switching to a competing service provider as they would lose the benefit of their existing provider's larger user base. For example, there is little incentive to be the first person to migrate to a new social network, leaving all your friends behind on the old one.

Two common policy requirements to tackle this concentration of digital services are service interoperability and support for multi-homing.

Interoperability between two service providers allows users to communicate across networks of users (e.g., between users of two different messaging applications). Mandated open interfaces, in the form of service interoperability, are a potential way to mitigate some consequences of the "network effect", tackle market concentration (for instance, in social media), and reduce the negative impact of switching. Such measures are often inspired by existing examples from industries such as telecommunications and include proposals to mandate interoperability across instant messaging systems.[12]

Multi-homing[13] allows people to use more than one network at the same time for identical or very similar services. In this instance, users might not be able to communicate across the two service providers' networks, but they can still benefit from porting their data to a competing service offering features or functionality that they prefer (e.g., porting a contact list from one application or service to another). Multi-homing is also of great importance on the supply side of platforms, for example, property owners being able to offer their holiday rentals both on Airbnb and competitors such as TripAdvisor.[14]

Addressing bottlenecks for innovation

The relation between interoperability and innovation has proved true in many domains—the strongest example being the Internet itself.[15]

In addition to data portability and service interoperability, other proposed policies include mandatory technical interfaces and mandatory data access and sharing among companies. Some policies require companies to provide open interfaces (including access to data) to allow interoperability between direct competitors and to enable innovative uses by new companies entering the market.

The EU has many existing policies to promote interoperability and access to data. For example, the public sector has mandatory open data and interoperability.[16] The 2017 Free Flow of Data Regulation supports portability of non-personal data between firms with a limited code of conduct[17]. Meanwhile under the EU Payment Services Directive 2, and Open Banking in the UK, banks can open up their data and services to third parties (strictly with the customer's consent). These third parties include banks with support for account switching, but also value-added services such as budgeting, insurance brokerage, etc.

11 For a useful overview of the concept of network effects, see http://oz.stern.nyu.edu/io/network.html

12 Stigler Committee on Digital Platforms (2019), Final Report, https://research.chicagobooth.edu/stigler/media/news/committee-on-digital-platforms-final-report

13 Crémer, de Montjoye & Schweitzer (2019), "Competition Policy for the digital era", European Commission, https://ec.europa.eu/competition/publications/reports/kd0419345enn.pdf

14 For further discussions on the concept of multi-homing, see: Park, Kyeonggook and Seamans (2018), "Multi-Homing and Platform Strategies: Historical Evidence from the US Newspaper Industry ", Harvard Business School Technology & Operations Mgt. Unit Working Paper No. 18-032, https://ssrn.com/abstract=3048348

15 Gasser, U. (2015), "Interoperability in the Digital Ecosystem", https://ssrn.com/abstract=2639210

16 See e.g., the European Interoperability Framework (EIF), https://ec.europa.eu/isa2/eif_en, and "The Directive on open data and the re-use of public sector information", https://ec.europa.eu/digital-single-market/en/european-legislation-reuse-public-sector-information

17 "...encouraging providers to develop codes of conduct regarding the conditions under which users can port data between cloud service providers and back into their own IT environments..." https://ec.europa.eu/digital-single-market/en/free-flow-non-personal-data

The recent draft EU Data Strategy for artificial intelligence and innovation aims to bring limited mandatory data sharing under fair, transparent, proportionate and/or non-discriminatory conditions. Proposals include a legislative framework, finance for pooling common European data spaces and opening commercially held datasets.

Key concepts and technical fundamentals

As noted above, the calls for open interfaces often focus on the desired outcome (e.g. "service interoperability", "data portability, and "data access"). The technical interface required to achieve such an outcome is often implicit, and one of the main aims of this paper is to discuss the associated technical considerations to help inform the debate. Below we define and introduce key technical concepts as used in this paper.

Defining "interface", "interoperability" and "implementation"

Open interfaces and interoperability are both familiar and well-explored concepts that have long been present in computing, networking and software. The terms are also frequently redefined, so people saying the same words might mean different things by them. As a consequence, the outcomes expected from mandating them vary depending on the assumed definitions.

Generally speaking, an **"interface"** refers to the point of contact between two software systems. This point of contact may be within a single computer (such as between the browser and the network software), between two connected computers, or over a network or the Internet. Or, indeed, it may be to an external device such as a printer, or to the human user. As a general rule, an interface introduces a layer of abstraction between a system's internal and external functions and structure. "User interfaces" and "application programming interfaces" (APIs) are both examples of this.

A user interface lets the user to perform defined actions without knowing how the system does what it is asked. For example, some cars now have an interface to their engine in the form of a "Start" button. The driver can start the car just by pressing this button, even if they have no idea of what sequence of electromechanical events that initiates. The start button is essentially an abstraction, shielding the driver from complexity and allowing them to interact more easily with the system.[18]

Similarly, an API enables a system to expose a consistent set of external functions to another system, even if how those functions are implemented changes internally[19]. This also applies to a system's data structures: the external interface can insulate users from the details of (and changes to) internal data structures.

The term **"open interface"** means that the interface specifications are, to varying degrees, made publicly available. The word "open" implies a similar worldview to open source software, where developers expect to be able to use, improve and share software in any form and for any purpose without *ex ante* or *post hoc* negotiation with rights holders. However, "open source" does not necessarily mean uncontrolled: open source software can still be covered by one of many licensing agreements which impose obligations or constraints on the developer.

"Interoperability" refers to the ability of different systems to seamlessly share information and resources. This is either when both systems must do something in order to complete a task (such as implementing both ends of a communication protocol) or when the output from one system must be usable by another (such as creating a document in one application and editing it in another).

18 Note that a traditional car key can equally be described as an interface since it is not obvious to most users what turning the key actually does to start the engine (closing the electric circuit between battery and starter motor).

19 This feature is especially useful in large scale systems as APIs can be used to create boundaries of stability, eliminating the need for applications utilizing this interface to be updated every time something changes "under the covers," which often happens frequently in complex applications.

Interoperability requires that either the two systems share the same **"implementation"**, meaning shared internal functions, logic and data structures, or that they expose interfaces which ensure that any differences in internal functions, logic and structures have no effect on their ability to interact, function and exchange data.

Maintaining interoperability can be challenging when systems have different internal functions, logic and structures, even if they expose open interfaces. The next section of this paper explores why this is the case.

The mechanics of software: data models and state machines

The **data model** and the **state machine** represent the inner workings of a specific implementation (i.e. software system). Since an interface's purpose is to enable interactions between different software systems, the data model and the state machine are central to understanding the opportunities and limitations of open interfaces, and the consequences of any mandates.

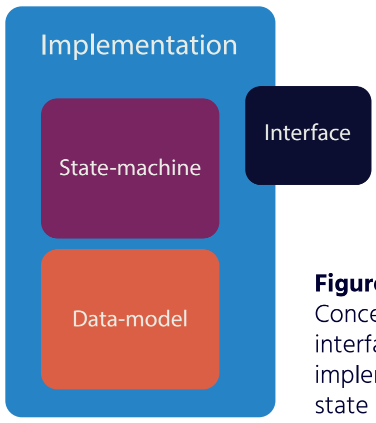

Figure 1.
Conceptual illustration of the interface as a point of contact to an implementation, and its associated state machine and data model.

The data model contains all the internal information that a system needs to perform its tasks.[20] For instance, a banking application must, as a minimum, have a data model capable of storing and processing the account number, account name, current balance, debit transactions, credit transactions, and so on. In some cases, the data model will have evolved from scratch with the software; in others it will have been developed to be compatible with the data models of other, pre-existing systems. In most cases, it will be a mix of both.

The system's state will change as it does its work, hence the relevance of the state machine: the state machine defines the sequence of states in which the software can rest. One way to think about it is like the sequence of traffic signals. In the UK, the set of red-amber-green lights can be in a limited range of states: red on, red and amber on, green on, amber on, all off. Moreover, these states can only be displayed in a fixed sequence (called "state transitions"). A full description of the system will include all its possible states, and the possible transitions between them.[21]

For any form of interoperability, be it via data import, network connections or programmatic exchange of data, a system's interfaces expose a defined

20 The design of the data structures may be elaborate and distinct enough for its authors to consider it an "intellectual property" asset in its own right.

21 For example, if the state is that only the amber signal is lit, the next state will be that only the red signal is lit. If red and amber are both lit, the next state will always be that the green signal is lit.

Similar software, different implementations: The example of word processors

The internal data structures are likely to be designed to reflect the way the word processor manipulates the document. There are many different approaches to manipulating documents. One approach is to treat the document as an optimal sequence of keystrokes or events to create the document. Another approach is to treat the document as a set of layers superimposed on an earlier version to produce the current version. Another approach is to describe the document in terms of its abstract layout and then apply a "style sheet" later that defines how to display or print the document. These descriptions are greatly simplified and there are many more approaches.

However, the result of this diversity is that there may well be data structures used by one word processor (and thus their file formats) which have no equivalent expression in the code of another word processor. For example, one very old word processor which worked a lot like a typewriter included the option to type a keystroke that resulted in the line containing that keystroke to be centred on the page, without affecting the adjacent lines even if they were part of the same paragraph. This effect has proved impossible to implement in many subsequent word processors, so when a file from that early system is loaded today, even assuming there is an import function for it, the "centre line" keystroke is typically discarded when encountered.

set of functions that remain consistent even if the way they are implemented internally changes over time, or differs from the way another system has implemented them.

The greater the internal differences between two systems, the less likely it is that their data models and state machines will be mutually understandable. This is clear in systems with complex outputs, such as a word processor. This stores not just the contents (text) of a document, but also details of the fonts and formatting that it uses, the history of how the document has recently changed, information about who is permitted to view or change it, and sometimes much more. All this information is stored in a set of internal data structures, which form part of the data model. When the document is saved, the resulting file will include the information in those internal structures, so that the full document (i.e. the text itself and all the associated metadata) can be reloaded in the future.

By providing a layer of abstraction between internal and external functions, interfaces allow implementations to be diverse. However, this diversity can mean that when data models are exposed for external access, they are transformed, rather than merely copied. This is especially the case if the external form is static over time, or even a standard. Transforming from an internal to external format could result in data loss, which the end user may then take to be a breakage—for example, in the case of the word processor, page breaks appearing in different places, or numbered lists losing their sequence. Where data is transformed there is also a risk that the transformation will not be perfectly reversible, resulting in a more subtle data loss when the data makes a round-trip between systems.

The same principles apply to state machines. In principle, a flow of events like a communications protocol can be represented as a sequence of states through which the system passes. In practice, two independent systems might each carry out the same sequence of states, but legitimately do different things between states. So, while both systems are implementing the same protocol, their state machines may well be different.[22]

Mapping from one design to another is therefore frequently an approximation, with interpretive compromises that can lead to display or function errors.

The interface in practice

As described above, an interface is the point of contact between two systems. What it looks like in practice depends on the desired outcome and includes a range of different options: from a simple export of data (without state), to those where a "broker" is used to facilitate the communication. Below we describe what those practical components consist of.

Data structure

An interface delivers the payload of a data structure—an element from a data model—between two programs. That data structure may be an independent specification, such as an open standard, or it may be documented less formally, such as in a "how to" guide or an existing open source implementation.

For example, the standard format of a .ics file means that it can be interpreted by many different calendar applications.[23] The X.509v3 specification for public key certificates means that multiple parties, using different products, can exchange and/or process certificates that they did not generate.[24]

Protocol

A protocol is the defined sequence of data structures and states exchanged between a sending and a receiving system. The structures exchanged (directly or indirectly) will be a subset of each program's data model, and the corresponding state transitions will be a subset of each program's state model.

22 To take a trivial example, imagine a server "polling" multiple sensor to get temperature readings. Both the server and the sensors are implementing the same protocol to talk to each other, but the sensor only needs to know two things: what's the temperature, and am I being polled? Its state machine is very simple. Meanwhile, the server needs to know lots of things, including: what are my sensors called; which one do I poll next; did I store the result successfully; what do I do if a sensor doesn't respond, and so on.

23 See https://tools.ietf.org/html/rfc5545

24 See https://tools.ietf.org/html/rfc3280

A very simple example of a protocol is in radio communication, when one person doesn't speak until the other person says "Over" (i.e. "Over to you"). If the sender says, for example:

> "Send help at once. Over."

The message is a data structure containing a payload and a protocol instruction. The payload is the phrase "Send help at once", and the word "over" is the protocol instruction telling the other party to switch from "send" state to "receive" state.

Application Programming Interfaces (APIs)

An Application Programming Interface (API) is a programming concept with various meanings. Traditionally it has referred to a functional interface of a program. Over time, it also came to mean an exposed functional interface that could be accessed dynamically at runtime. In both cases, the software involved would be run on a single computer. Simple, stateless export of data would fall into this category.

When networked access became the norm, "API" was reinterpreted to also include exposing interfaces to the running code for use by other applications across a network. Distributed processing gave rise to the need for mutually understood states between communicating partners—and this is where protocols enter the picture.

Early examples were described as remote procedure calls (RPC), with markup syntax gradually used to standardize their payloads (XML-RPC). Today, generic text interfaces are used to inject messages into a running program. These are then parsed and the data structure payload extracted, analysed and passed to an appropriate internal interface. An interface of this type is sometimes described as a "socket interface".

The next step is to treat each request as a self-contained instruction. Here the server does not keep track of what state a client is in, so there is no need for a synchronised protocol between the two. The data is most often delivered using a "Representational State Transfer" (REST) interface[25] (or REST API) using a message formatted in Javascript Object Notation (JSON), according to the OpenAPI standard.[26]

Another option is the message broker architecture.[27] Here, distributed applications can access load balancing and other services via a component (the "broker") whose only function is to receive "layered" messages and then pass their payload to other services. A payload is wrapped in one or more layers, each of which identifies the service that can process the contents.

Message brokering is a modern successor to the earlier object brokering approaches embodied in XML-RPC and SOAP.[28]

25 See https://www.codecademy.com/articles/what-is-rest
26 See https://www.openapis.org
27 See https://www.ibm.com/cloud/learn/message-brokers
28 See https://www.w3.org/TR/soap12-part0

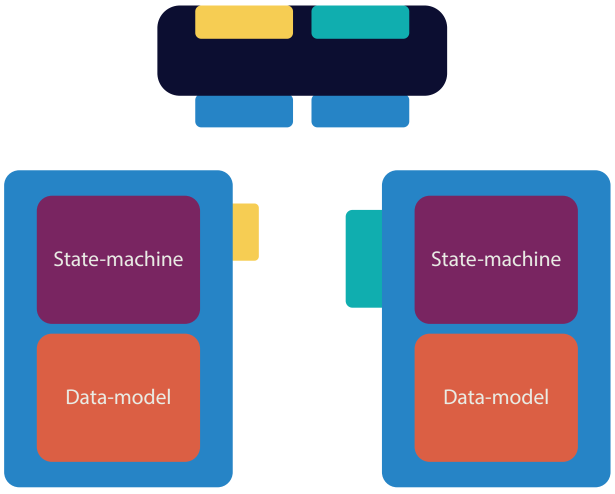

Figure 2.
Conceptual illustration of a message broker as an intermediary point of contact between two implementations.

By default, message broker architectures create the option of abstraction. That is, by separating components and providing a generic request interface between them, a broker architecture makes it easy to introduce functions such as translation, reformatting, load balancing, and so on, between the client and the service it requests.

Thus, the API that's visible to a client may not be the actual interface to the service in question, since that is behind translators and load balancers whose job is to translate an external API to an internal one. This "translation" can give the impression that an open interface is present, even if one or other party is not using an open request format.

Practical considerations

Successfully mandating service interoperability, data access and data portability requires that there be a technical interface. In some cases, a mandate has succeeded without explicitly defining such an interface—but in our view, better understanding of the role (and possible limitations) of the interface will lead to better-defined mandates. Therefore, in this section we highlight the practical considerations that might affect this interface, from its conception to its operation.

Technical aspects

First, we need to look at how to design a mandated interface—for example, how would it need to be framed to achieve its intended purpose? The first challenge is to work out the mandate's purpose in specific and tangible terms, including intended outcomes, the locus of control and the deliverable interface.

Intended outcome

The type of open interface most appropriate to a given situation will depend on the intended outcomes. Some examples are:

- to ensure that customers can switch service providers with as little hindrance (or friction) as possible;
- to enable one service to use the data generated by another;
- to allow several services to use the same state and data (for example, synchronizing contact books on different devices used by the same user);
- to enable users to collaborate (e.g. real-time messaging) across different services.

Service mobility

The simplest level is service mobility—the ability to switch between two service providers who apparently offer the same service. If they actually offer the same service (and do so by running different instances of the same software), they will share implementations, meaning the switch should be straightforward. This is one of the benefits of using open source software, as various parties collaborating on a shared codebase can reduce their interoperability costs. An example of this in practice is libc, the standard library for the C programming language.[29]

29 See https://directory.fsf.org/wiki/Libc.

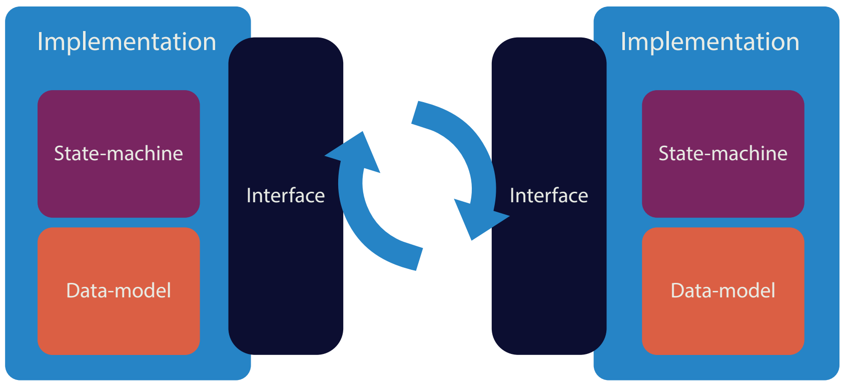

Figure 3.
Conceptual illustration of how having identical implementations enable the use of an interface that can expose a much wider set of functionalities.

However, when services are running different software (including different versions of the same software) interoperability can be a challenge. One-time data export from the source system may be possible, but the differences between data models and state machines may result in only a partial importation, with some information lost, incorrectly transformed or lacking access to external data or systems that make it useful. Repeated transformations, due to re-exporting the data and "round-tripping" it back to the original system, may produce even more anomalies, as approximate mappings in each direction introduce slight incremental changes, like the children's game 'Telephone'.[30]

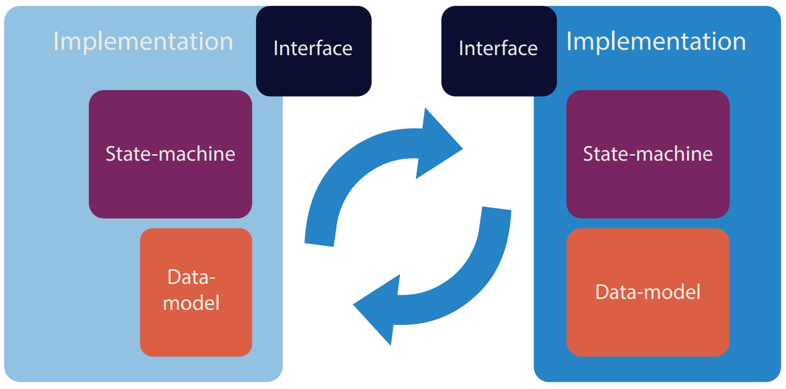

Figure 4.
Conceptual illustration of how having different implementations implies that a smaller set of functionalities can be exposed in the interface, and anomalies may result in breakage.

A data model as complex and nuanced as the one representing a social media system like Facebook or Twitter gives us a helpful example to think about the challenges involved. For instance, exporting a subset of the social graph that denotes a user's unweighted associations with different objects (e.g. users, photos, pages, etc.) is a relatively straightforward task. But more nuanced or complex elements of the social graphs, such as indirect associations, their calculated weight and more elaborate metadata, are probably difficult to export, even as a snapshot, in a way that allows an alternative service to offer identical functions. In part, this is due to the fact that the data model itself may be only part of the picture of providing the full functionality. Internal systems might manipulate the data in ways which are essential to support the service's full range of functionality, but which are not revealed as part of the export, and which cannot be inferred from the data itself.

Shared data and state

Where systems are interoperating to carry out distributed processing, they will need to exchange data and/or state on demand. The simplest approach to this is to provide an interface through which services can share their data and/or state so that both can process the same information in a similar manner.

30 In the game, participants form a line and a message is whispered from one participant to the next until it reaches the end of the line. The first person then compares the original message with the one that reached the end of the line. While the objective is to pass the message without changing it, the final version is usually completely different—which in this case is part of the enjoyment.

An example is the UK's Open Banking,[31] where banks provide an interface that lets financial service providers, such as accounting software systems, access account data for their customers "live" (i.e. in real time), rather than needing to manually import transactions, which would lead to delays.

Since applications written by different authors can be expected to have different data and state models, providing shared access will almost always involve subsets of functionality. Subsetting may involve the shared use of the same data models, or more likely will be achieved via a data structure transformation, as discussed above. The subsetting of functionality is perhaps the greatest technical barrier to true multi-homing.

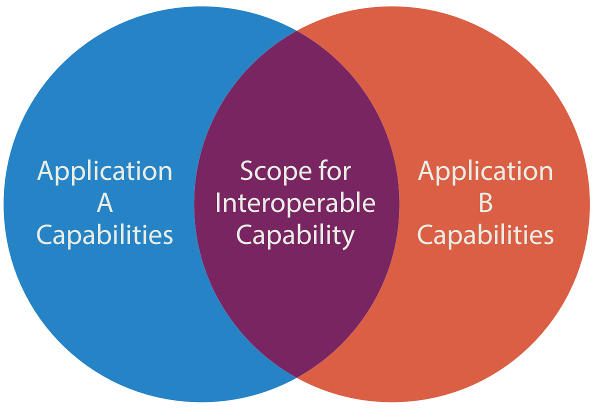

Figure 5.
Illustration of the interoperable capability as a subset of functionality.

Inter-service collaboration

A more advanced approach enables collaboration between services, such as real-time messaging. For this to work, systems need both compatible data structures and some form of compatible state transition. In practice, the two systems must either have synchronized, coherent transitions of state (i.e. a mutually implemented protocol) or settle for just those transaction types that can be satisfied with the simpler REST-style approach to handling state.

For example, in the case of Open Banking, the interoperating parties agree on a fixed subset of "transactions" to achieve limited interoperability between multiple service providers. Basic account details, such as balances and raw credit and debit transactions are retained, but more advanced information, such as budgetary reserves or anticipated future transactions, are not included.

This subsetting can be challenging, but for simple systems like instant messaging, a pluggable transform approach such as that used by Pidgin/libpurple[32] might also help to increase the range of interoperable capabilities. Even for this apparently simple case, a great deal of diversity and incompatibility has evolved in the market.[33] Appendix A's "family tree" of instant messaging protocols shows the complexity that can emerge even in open interfaces.

Locus of control

In this section we look at where interoperability needs to be achieved and ask a number of questions. For example, who owns and/or controls the state machine and the data? Is the goal to make it easier for the user to switch from one application, service or provider to another? Or is the goal to create interoperability between peer services? Does that extend as far as enabling users of different services to collaborate? Is this collaboration in real time, or by sharing resources, or round-tripping, or just by requesting portability via standard file formats?

31 See https://www.openbanking.org.uk
32 See https://developer.pidgin.im/wiki/WhatIsLibpurple
33 Vaughan-Nichols, S. (2017),"The great instant-messaging foul-up", ZDNet, https://www.zdnet.com/article/the-great-instant-messaging-foul-up

Substitutability

The most fundamental level of openness is when an end user is able to substitute one service for another, giving the "service mobility" outcome above. The result of this transition is a change of "ownership" or control of the customer's dataset from one service provider to the other. Substitutability may be a one-time export process that's only available on request (like a mobile phone Porting Authorisation Code (PAC)[34]), or users may have the freedom to access all data at any time.

A practical and widespread example would be the open e-mail standard Internet Message Access Protocol (IMAP)[35], since you can read and send mail from your phone and from your desktop client without regard to which you are using.

With substitutability, the new service provider will be able to take full control of the data and state related to the user. This could involve the data passing first to the user, or it could involve the previous and the new service providers communicating directly with each other (as in the phone PAC case). Both approaches will have their own considerations relating to proving identity (including both authentication and authorization), maintaining backups and managing the relationship with the other parties.

Interoperability

Interoperability is where two users with different software can independently share data, or use the same service, without either needing to migrate to different software. It also describes the case where a user with two different systems can connect to the same data in both and—in cases of true compatibility—create the same state. In both cases the original service provider is likely to be the primary locus of control for the data and associated state machine.

There are also arbitration considerations, in case of a disagreement between interoperating systems. For example, if two different instances are working on the same document, one may overwrite the other—how is such a 'conflict' resolved? Another thing to consider is how to identify different users and the levels of authorization that various identities may have.

In other words, interoperability is practically certain to require much than just both parties adopting an open interface—it also needs a common protocol and a sufficient degree of shared implementations.

Collaboration

Collaboration is when two users of interoperable software can work on the same task at the same time productively, such as when two users collaborate on editing a word processing document. This case is especially complex, as both applications would need to mirror the data and state of the shared activity, while independently manipulating the subject matter.

In virtually every usable case of collaboration to date, a third application (considered server software) is needed to control the data and state and to synchronize with the two users' applications, or to offer a location for the actual collaboration.

34 For more information about PAC, see https://www.ofcom.org.uk/phones-telecoms-and-internet/advice-for-consumers/costs-and-billing/switching/switching-mobile-phone-provider

35 See https://www.makeuseof.com/tag/pop-vs-imap/

How much interoperability is enough?

Early reviewers of this paper noted that some sections seemed to imply only 100% functional equivalence and interoperability would be acceptable, and told us that much smaller percentages may be perfectly adequate. We agree that in many instances this is likely to be the case, but it raises the question of how much interoperability is enough interoperability? The answer is "it depends". Below we illustrate this with two examples:

Use-case demands

Take for example interoperability between two document processing systems. Users who are drafting blocks of text that will be used in formatting-insensitive ways—within a system with stylesheets, or pasted into a larger work, or for processing by other software, for example—will find that almost any degree of interoperability that preserves the plain text when accessed on the interoperable system will be sufficient.

Users creating the near-final version of a document whose formatting is critical—for example a legal document for court usage where exact line numbers and their contents need to be precisely reproduced on every system, or generating the camera-ready proofs of a document to be reproduced by offset lithography—will demand that every single detail of the document they produce in their software be precisely and faithfully reproduced when loaded into a different software system. Truly, nothing less than 100% is enough. For some, even variations in font kerning would be a fatal flaw.

The same system, the same standard, but with different use-cases, will have different expectations and requirements for how much interoperability is enough. A dominant vendor might well implement capabilities on the margins of a standard in order to cause import differences by a competitor's software that the user will then blame on the competitor. In addition, issues that are perceived as interoperability issues by the user may actually be due to platform[36] differences. For example, many issues reported by LibreOffice users actually arise from the availability of the correct fonts on their platform since the default Windows fonts are not universally available on other platforms.[37]

Given the difficulty of delivering a user experience perceived as identical on alternative implementations, how has software like Google Docs been able to gain such a large user base? Instead of a focus of a substitutable user experience, Google started with a compelling new capability—real-time user collaboration and change tracking—and implemented good-enough interoperability using open source tools.

Implementations may affect the interoperable experience

Even where the capability involved is a subset of function rather than the whole system, the use case will control the level of interoperability that's acceptable. Consider a larger system intended for some other purpose—social media, forums, a phonewith the addition of integrated text messaging that can receive messages from other people. For some users, the only function that matters is sending a short, text-only message to another person and having the text faithfully reproduced.

Another user may want to include emojis in their message and expect the same emoji to be seen by the recipient—it's almost the same use-case. If the receiving system displays from a different emoji palette, using only the image definitions in the standard for reference, it's possible that entirely the wrong impression could be given.

A quirky, yet real, example is that many users treat the "folded hands" emoji (U+1F64F) as a "high five" while some treat it as "applause" and many others treat it as "praying hands". So, the message "Just heard your news 🙏🏽" could be read as a concluding celebration, an empathetic prayer or as personal congratulations depending on the reader and their device[38], despite full conformance with the standard and apparent 100% interoperability.

Naturally, such a scenario is inherently about cultural semantics, and while any impact *from variations in the design of the emoji* could, in theory, be avoided by requiring that the implementers make use of the same emoji palette, the example illustrates that things outside of the specification can include nuances that affect the interoperable experience, and that are hard to predict. It is also a reminder that an inherent feature of standardization is to enable variability—in this case, the ability to differentiate your service through emoji design while conforming to a standard.

36 See https://design.blog.documentfoundation.org/2016/10/21/dealing-with-missing-fonts/
37 See https://www.onmsft.com/how-to/how-to-install-microsoft-fonts-on-linux
38 See https://blog.emojipedia.org/emojiology-folded-hands/

Developing the interface

This section looks at the process of creating and developing an interface, as well as licensing and rights.

Some might suggest the answer to the above questions are "use open standards" or "just use open source". But those both come with their own considerations. What is an "open" standard in relation to any other *de jure* standard, for example? Who gets to define it and on what authority? How long does that take? Is a reference implementation the same as an open project? How can open source be used most effectively?

Shared interfaces and approaches to standards

Like "interoperability", the simple word "standards" conceals a plethora of different approaches to agreeing on interfaces between entities. These approaches mainly fall into two large categories—requirements-led and implementation-led[39]—and each category implies a range of expectations about how agreement is reached, who can use the standard and for what, and who needs to pay for usage (if required under the license) and on what terms.

While they have much common terminology, the sequence of innovation and standardization in the two categories is very different, which can lead to misunderstood expectations and unintended consequences. Successful openness mandates must address these differences, and not simply refer to "standards".

A **requirements-led** model is used by industries where high degrees of interoperability are essential to market formation and operation, and require *de jure* standardization—such as the mobile phone industry.

In this approach, an industry forum (for example, ETSI—the European Telecommunications Standards Institute[40]) acts as the location for deployers to express their requirements. Suppliers then propose responses to these requirements, declaring in advance any patent interest they may have in their proposals. The industry forum then evaluates the proposals and selects options at each function point to build a specification, accepting the patents disclosed in the process as "standard essential". The specification is ratified as a standard, and a market forms in which suppliers and deployers trade. The suppliers whose proposals were included in the standard are then free to recover their costs and profit from all implementations of that standard, based on their "Standard-Essential Patents" (SEPs). The industry forum will usually require FRAND[41] licensing terms, but often each deployer will also need to negotiate over such terms, which are rarely public or uniform.

In the alternative **implementation-led** model, an early market entrant or market leader innovates privately and brings a new product to market. This product will expose some form of open interface and over time partners will implement new capabilities using that exposed interface, turning it into a de facto standard. As the market emerges and competitors begin to reimplement the interface, the need for *de jure*[42] standardization arises. An industry forum acts as the location for various market stakeholders to discuss the canonical form of the interface, and a specification is collaboratively written that defines its function. This specification is then ratified as a standard and gradually the existing market conforms to it, leading to new scope for competitive behaviour and innovation.

In the implementation-led model there is no incentive for patent monetization, so standardization activity is either under mandatory RF (royalty-free, or sometimes restriction-

39 Phipps, S. (2019), "Open Source and FRAND: Why Legal Issues Are The Wrong Lens", Open Forum Academy, http://www.openforumeurope.org/wp-content/uploads/2019/03/OFA_-_Opinion_Paper_-_Simon_Phipps_-_OSS_and_FRAND.pdf

40 See https://www.etsi.org

41 While the term "FRAND" arose as an acronym of "fair, reasonable and non-discriminatory" it has gained a life of its own. In essence it is usually taken to mean a commitment by rights-holders not to profiteer around IP control points, but the term's vagueness has led to lengthy disputes internationally. Notably, patent holders largely reject ex ante terms and expect to be able to negotiate one-to-one with each implementer. Whatever the definition, it thus means that negotiating terms will be required post hoc if not before. This, and not the licensing price, is the key to understanding why "FRAND" and "open" are controversial.

42 In an EU context people often talk about formal standards or mandated standards—standards as part of or resulting from directives and regulations. That is not the meaning of "de jure" in this context. Here we talk about de jure standards when they are developed in any, formal or not, standards organization.

free) terms (for example, W3C—World Wide Web Consortium[43]) or the stakeholders mutually self-select RF terms from multiple options (for example, OASIS—Organization for the Advancement of Structured Information Standards[44]). As a consequence, the term "SEP" is largely absent, as it is irrelevant.[45]

Anyone aiming to mandate open interfaces could specify that they *must* be implemented as open source. This would be a way to require openness without committing to a specific model. The Open Source Initiative has usefully created the Open Standards Requirement for Software,[46] which identifies which approaches to standardization can be implemented as open source software and which ones cannot, and an alliance of *de jure* standards organizations has also created a unified approach to open standards.[47]

Shared implementations and open source

One way to make interoperability more likely within some functional subset is to encourage implementers to use the same source code for the parts of their application that handle the interoperable functions, so that both the state machine and data model are actually the same at both ends. An interface that is made generally available like this can arise from a reference implementation of the standard—created by the parties who wrote the specification—or it can involve an independent and original implementation that is made freely available. Both will involve open source licensing.[48] The rights involved in the specification or market-making work—especially patent rights—must thus be licensed in a way that makes open source implementation feasible.

The term "open source" refers to software which has been published with a license that allows others to use and improve the software, without negotiation with the rights holder. At one end of the range of open source licensing options, software may be available to be shared, with no further obligations. At the other end of the range, various obligations may apply, which may or may not conflict with the users business model. Some stakeholders will argue that the ethos of open source is based on the assumption that rights holders will want to gravitate to the "unrestricted" end of the spectrum—but even among open source advocates this can be a bone of contention. In any event, when choosing an open source package, one should check that the terms of the license fit with its intended use, with just as much care as for any other license agreement.

The term can also refer to a corresponding network effect where, confident of their freedom both to innovate and to pool their work without the need for further negotiation, developers work collectively around the same code-base and attract others to do the same. Both are commonly achieved by the use of a rights license approved by the Open Source Initiative[49] as conforming to the Open Source Definition.[50]

Note that simply being open source doesn't imply anything about software quality or security. One of the advantages with open source is the opportunity to have an extensive set of reviewers, but just because software is made available to be reviewed and evaluated doesn't mean that this has actually happened. The skillsets required to perform this kind of work at the high level required can be specialized and may thus require the involvement of experts with the proper knowledge and background.

Granting rights—FRAND and rights waivers

For an open interface to be feasible, the rights needed to implement both it and the software that serves it must be made available. This especially applies to intellectual property rights, which will affect global adoption even if they prove unenforceable in individual jurisdictions.

43 See https://www.w3.org
44 See https://www.oasis-open.org
45 The Internet Engineering Task Force cannot be qualified as being at either end of this spectrum. Its processes allow for patented technologies; however, a lot of the standards resulting from the process can be used on royalty free terms. For more information see https://tools.ietf.org/html/bcp79
46 See https://opensource.org/osr
47 See https://open-stand.org
48 See https://opensource.org/licenses
49 See https://opensource.org
50 The Open Source Definition is itself derived from the Debian Free Software Guidelines. See https://opensource.org/osd

Every standards body has policies that ensure that rights holders must declare patent interests, but the terms vary significantly. Most today require a "FRAND commitment" from participants, while some (e.g. OASIS) go further to allow certain working groups to specify royalty-free (RF) use of standard-essential patents, and others (e.g. W3C) apply an RF requirement to all work.

Mandating open interfaces will require at least an RF use for any *de jure* standards[51] to ensure there is equal opportunity to implement.

Where a *de facto* standard is involved, implementers of a mandated interface will want to know that the most likely rights holders have provided rights.[52] This can be achieved with some form of rights waiver, such as the Google Open Patent Non-Assertion Pledge[53] and many others.[54]

Operating the interface

Since the open interface provides a point of contact between software systems it is important, from the outset, to consider its ongoing operation and ensure that it can achieve its intended outcome. This includes use and access policies, and other operational aspects that could affect its secure and reliable use, and which may need to adapt to meet the changing needs of the interoperating parties.

Managing change and use

While it may be easier for policymakers to take outcomes as their starting point (as opposed to, say, technical implementation details), it is also important to allow for the internal realities and practicalities of the platform implementer and operator. Poor choices in an open interface mandate could lead to large differences in the cost, risk, and even feasibility of compliance —resulting in solutions varying greatly for externally imperceptible reasons. In addition, operational considerations can be "weaponized" to artificially restrict competitors.

Change control policies

It's easy to think of an open interface as a public resource, but it remains the responsibility of the entity that continues to manage it. Unrelated parties are likely to become dependent on it, which raises issues of integrity and change control. Who can require changes in the production system, and under what circumstances? Do they need to follow a published (or even public) process?

An interface will evolve over time, as the underlying standard matures or as market conditions change. Changes to an interface may trigger considerable rework by those who use it. When changes are made, such as adding or removing parameters, or even deprecating capabilities, who must be told, and when, and by what means? How much needs to be disclosed about the change and the reasons behind it?

Changes to an interface may well require changes to the architecture of the systems they enable, which could be significant, costly and time consuming. When changes are made, does there need to be a support mechanism for users, or even an appeals process?

It is often expected that an interface will be changed alongside its earlier form, and deprecated parameters and data structures will remain available for some time, providing backwards compatibility after the new versions are issued. Note however that there are cases in which you want to intentionally break backwards compatibility, in order to force a change, for example to eliminate something which was in wide use at one time but which was later found to be easy to compromise. What backwards compatibility capabilities should exist after changes, and for how

51 Such a requirement may not be met if the stakeholders involved in developing a standard are not aware of essential patents that exist for the technology.

52 This is the best that is probably achievable in a world where non-practicing entities (aka "patent trolls") remain free to acquire distressed patent assets and use them to blackmail genuine businesses. The fact that it is impossible to guarantee patent safety from unexpected attacks should not detract from the need to mandate rights grants from the most likely holders.

53 See https://www.google.com/patents/opnpledge/pledge

54 See http://xml.coverpages.org/ni2005-10-04-a.html#patentCommons

long? Conversely, would any backwards compatibility measures be undesirable—for example, if they led to a violation of a subsequently-introduced data protection law?

Rate & access limits

An open interface is a gateway to the systems that service it, and increased use will increase processing burdens. As such, it could enable denial-of-service attacks, where a malicious actor tries to overwhelm the system with a tidal wave of apparently valid requests to process.

Interface providers commonly restrict access to mitigate this risk, for instance by supplying an "access token" in calls to the interface. Access tokens allow the system to distinguish between users, and to apply usage limits to protect it from abuse. A rate limit, which restricts how often the interface may be used, is the most common usage limit. However, since the interface provider could also use rate limits to restrict competitors, a mandate may need to define "reasonableness" tests for any operational limits, such as how often an interface may be used, or what scope of data can be requested. This may be particularly important if the reason for mandating the open interface was to prevent anti-competitive behaviour by the interface provider.

Managing access

As noted in the beginning of this paper, the very essence of an open interface is to define and enable a well understood point of contact between two systems. While the nature of that contact varies, from one-time exports of data to ongoing inter-service collaboration, key questions around access to the interface must be answered to ensure controlled, reliable, and secure communications.

Identity management (authentication and authorization)

Identity is the most common control on systems, in terms of authentication (who is this user and how can they prove it?), authorization (what roles and privileges does this user have?), and access control (to what resources do those roles and privileges grant access?).

In any distributed application, the owner of a resource is being asked to grant access to it, based on credentials and privileges they almost certainly don't control. The baseline assumption is that one user can rely safely on the credentials issued by another, to grant access to its systems and resources. Interoperability therefore needs to encompass not just the technical mechanisms, but also the liability and contractual issues that could arise if a credential is wrongly issued or misused, or if access is granted when it shouldn't be.

Identity systems are critical control points in IT systems, so any open interface mandate will need carefully to consider the consequences of granting control over them. This is especially important in two respects, because of the distributed (and often federated) nature of so many online services.

First, for controlled access to the interface itself, how can users of the open interface effectively identify themselves? As explained above, it is common to use an access key as a parameter, but even static access keys need a mechanism for request and management (including revocation). How do these work? How do implementers manage the identity to which access keys are linked? Is it under the control of the interface supplier, or from some other source, such as an OAUTH[55] or SAML[56] provider? For example, the API key used by Twitter uses OAUTH[57] for authentication.[58]

Second—and particularly for social media systems—when it comes to accessing data through the interface, how are the parties involved identified? Can interface users access social graphs, and if so can they interpret their contents or even go link by link to learn about other individuals? In the case of social media, access to information about others is likely to depend on the rights that the subjects of the social graph themselves have to access information.

55 For more information and explanations, see https://www.csoonline.com/article/3216404/what-is-oauth-how-the-open-authorization-framework-works.html

56 For more information and explanations, see https://www.csoonline.com/article/3232355/what-is-saml-how-it-works-and-how-it-enables-single-sign-on.html

57 See https://developer.twitter.com/en/docs/authentication/oauth-1-0a

58 See https://developer.twitter.com/en/docs/authentication/overview

Even then, the resulting matrix of access rights might itself disclose information about the user, so to what degree and under what terms must these matrices be exported/shared/accessible? How are correlations between the users of different systems managed? More generally, how is the information-access privacy policy expressed and managed, especially contractually? How are records tracked, how are retention periods set and enforced, and how is access to them controlled after retrieval?

In both cases, matters of policy over the life cycle of credentials may also have an impact. This includes the need to determine how to create and renew credentials, how expired credentials are removed, and how compromised credentials are invalidated.

Key management

It is likely that public key cryptography[59] will be involved in each of these operational issues. Certain data structures may be encrypted; documents and emails may be signed or encrypted; messages in instant messaging systems may use end-to-end encryption such as OTR[60] or Signal protocol[61] to ensure privacy.

In each case key pairs will be associated with each author, user or end point of the connection. Those key pairs consist of a public portion that may be stored in a directory or a key server, and a private portion that should be carefully protected, with guarantees to the key owner. When cryptography is used, how these keys are stored and related access controls must be clearly defined. Who has access—the users, the service provider, the interface user, or a combination of the three? How are keys revoked and replaced, and under what circumstances?

Once keys are issued they are usually impossible to regenerate, so it is critical to have either a backup process or a procedure for replacing keys whenever necessary. Can keys be exported for backup, or indeed for use in another system? More technical users may already have public-private key pairs that are part of a web of trust[62], such as PGP/GPG keys.[63] Can existing keys generated elsewhere be granted trust, and how? How does key management operate across the interface and between interoperable systems? Generally, how do keys become trusted, and how and when does trust expire or get revoked?

There has been much talk of state actors requiring (or at least demanding) privileged access to the decrypted content of encrypted communications. This is of course impossible without compromising the security of the system involved, so it will be important to clarify whether an interface mandate requires that keys must be shared or intermediated.

Are there any jurisdictional issues? For example, some jurisdictions may forbid citizens of certain other countries from using certain forms of cryptography.[64] There may be a choice of cryptographic algorithms, both as a consequence of jurisdictional considerations and the evolution of systems and markets. Are alternative algorithms in use, and if so, how is one chosen or preferred?

59 See https://ssd.eff.org/en/module/deep-dive-end-end-encryption-how-do-public-key-encryption-systems-work

60 See https://otr.cypherpunks.ca

61 See https://signal.org/docs

62 See https://www.rubin.ch/pgp/weboftrust.en.html

63 This is itself a huge and complex subject. See https://www.openpgp.org/about/history

64 The US continues to treat cryptography as a munition in some circumstances. See https://www.bis.doc.gov/index.php/policy-guidance/encryption

Policy considerations

Beyond the fundamental issues of compatibility and how the interface operates are various considerations about public policy objectives. This section will outline some of these issues.

The impact on market dynamics

Mandating an open interface is often accompanied by a goal to change market dynamics to the benefit of consumers or smaller companies. How effective such measures will be depends on assumptions about current market conditions and the potential for unintended consequences as new incentives are introduced. Here we consider some of the questions that might arise when mitigating key challenges and look at the impact on a broader set of objectives.

Assumptions about superior products and the effect on startups

The narrative around competition and consumer lock-in is centred on the idea that better digital services may be available, but choice is restricted because of various technical and market barriers. However, in many cases the dominant product may be of a higher quality and be attractive enough to stop alternatives. In a virtuous cycle, a dominant supplier will have the resources and motivation to keep marketing a superior-quality product. In that respect, even full interoperability may not be enough to entice the average consumer to a competitive offering.

Competition policy in digital markets also tends to make assumptions about startups—mainly that smaller companies that develop innovative and better products are then bought by large tech companies in order to prevent the creation of a future competitor. The innovations are either incorporated in the offerings of the established outfits, or simply killed, leading to a reduction of consumer choice.

How open interfaces and interoperability work with these dynamics, and whether more integration can weaken independent efforts to create complete alternatives or lead to easier acquisitions and consolidate the market even further, is unclear. Intuitively, if emerging competitors to a dominant incumbent have to conform to a mandated interface, it is easier for the incumbent to acquire them and "swallow" their products and data. The narrative, in this form at least, also glosses over the effect that large companies' research and development programs can have on competition, especially through patenting potential innovations.

Will mandating an interface competitively disadvantage any market players?

A mandated interface could be used as a competitive tool beyond its intended purpose. For example, Facebook and Twitter, have faced complaints from third-party application developers that they abused control over access to their interfaces, and that this restricted access to user data not primarily to protect user privacy, but selectively to fight competitors.[65] In principle, a mandated open interface should solve this issue, but companies under competitive pressure

65 Solon, O & Farivar, C (2019), "Mark Zuckerberg leveraged Facebook user data to fight rivals and help friends, leaked documents show", NBC News, https://www.nbcnews.com/tech/social-media/mark-zuckerberg-leveraged-facebook-user-data-fight-rivals-help-friends-n994706

can be surprisingly creative, and could manipulate their product design, or their change control policies, to create *de facto* restrictions without explicitly blocking competitors.

Data sharing, pooling and interoperability could also create anti-competitive cartel effects[66] if their effect is to restrict other entrants' ability to gain access.[67] The EU takes the risk of cartels very seriously and has developed extensive guidance on what is termed as horizontal co-operation among competing firms, including data exchanges.[68] These tensions between competition and collaboration have not been resolved yet[69], but if the goal is to ensure that no market participant is ever excluded, one should not underestimate the degree of openness and inclusivity this could require, throughout implementation.

Interoperability does come at a cost, which may be disproportionate in the case of smaller market players. As a consequence, commercial pressures for and against data portability may not reflect "normal" market dynamics, and may raise the question of how to set and enforce the obligations of different stakeholders.[70] This is complicated by the fact that technology allows new platforms to accumulate mass quantities of data at a rate that exceeds their ability to govern it. If such companies also have to comply with data portability regulations, the overall administrative burden may be unsustainable for smaller players.

Another concern when mandating interoperability in markets that are dominated by an incumbent is that it could result in even higher competitive barriers around anything that is not covered by standards or agreements. In such a scenario any new companies entering the market would face burdensome compliance requirements dictated by the dominant players, effectively resulting in "everything that's not forbidden is mandatory".[71] Interoperability rules would thus become the ceiling, rather than the basis for new competition.

Safeguarding benefits to end users

Interoperability, portability and open interfaces may generally be good for consumers, but in some cases, data pooling and interoperability can create more power for businesses instead. In addition to the privacy problems caused by de-compartmentalizing separate datasets, consumers could get a worse "deal" based on the enhanced information that's available to companies. For example, in the UK all traffic accident claims data is pooled among insurance companies[72] and any incident when the driver was not at fault, or not even in the car, will be used to increase premiums. This may be legal and "fair", being based on data and statistics, but it is nevertheless detrimental for the individuals affected.

Importantly, and given the difficulty of clearly distinguishing between personal and non-personal data, this raises the question of what rights users have to control their data under a mandated data sharing regime. The European consumer network BEUC advises caution on mandatory data access, given that such a right to object to the sharing of data does not exist under the GDPR.[73]

From policy to practice

Translating high-level public policy objectives into the practical implementation of a mandated interface requires careful thought about the development of new standards, their legal status, and the disclosure requirements of existing mechanisms. As outlined previously, open interfaces

66 Lundqvist, Bjorn, (2018), "Data Collaboration, Pooling and Hoarding under Competition Law", Stockholm University Research Paper No. 61, https://ssrn.com/abstract=3278578 or http://dx.doi.org/10.2139/ssrn.3278578

67 Crémer, de Montjoye & Schweitzer (2019), "Competition Policy for the digital era", European Commission, https://ec.europa.eu/competition/publications/reports/kd0419345enn.pdf

68 See https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52011XC0114(04)&from=EN

69 See Wilson, T. (2017), "Cooperation and information exchange: innovating in a connected world", Kluwer Competition Law Blog, http://competitionlawblog.kluwercompetitionlaw.com/2017/11/17/cooperation-information-exchange-innovating-connected-world

70 Swire, P. & Lagos, Y. (2013), "Why the Right to Data Portability Likely Reduces Consumer Welfare: Antitrust and Privacy Critique", Maryland Law Review 335 (2013), Ohio State Public Law Working Paper 204, https://ssrn.com/abstract=2159157

71 Doctorow, C. (2019), "Interoperability and Privacy: Squaring the Circle", Electronic Frontier Foundation (EFF) blog, https://www.eff.org/deeplinks/2019/08/interoperability-and-privacy-squaring-circle

72 See https://www.mib.org.uk/managing-insurance-data/the-motor-insurance-database-mid

73 BEUC (2019), "Access to consumers' data in the digital economy", https://www.beuc.eu/publications/access-consumers-data-digital-economy/html

are an ongoing process that may require updates to technical specifications and changes to operational policies. A mandated interface is highly likely to give rise to other, non-technical tasks which are essential, even if they are not mandated—and without which the mandated interface may fail to produce the intended outcome. In other words, the (apparently purely technical) step of mandating an interface requires robust, transparent and efficient governance.

What should governments mandate for the best results?

Standards are a traditional cornerstone of interoperability. There have been proposals in the US to increase the regulators' powers to impose mandatory open standards that promote competition.[74] This can be a very effective way to overcome business blocks, but there are also issues around costs, lack of flexibility and the difficulties for governments to make the right technical decision.[75]

EU policy favours formal European standards, set through a process of collaboration between EU bodies and standard-setting organizations.[76] The EU approach is based on the principle that standards are voluntary, but that conformance may be required, for compliance with some EU legislation. Recent policy discussions have looked at voluntary standards for protocol interoperability,[77] but whether the availability of (open) standards is enough to ensure interoperability is unclear.[78]

Other options include licensing products or services, and for the licensing process to require that product information be made accessible to other participants (including actual and potential competitors). The licensing approach is evident in safety-related products, such as electrical power supplies for computers, or radio frequency interference. However, interoperability does not follow as a matter of course from this approach, as it depends on what product information is disclosed. Another alternative is to use certification rather than licensing: again, the process can be set up to test and assert the interoperability characteristics of a product or service, so buyers know that, if they buy product X, it is interoperable with product Y.

In both cases, a third party (the licensing or certification body) is assuming some level of responsibility (and potential liability) for validating the vendor's claims about the product—but whether or not that translates into interoperability depends on the nature of the claims and the extent of the validation.

Who is responsible for reviewing requirements, how often and what recourse do they have?

The governance of open interfaces is a challenge. Although the process may be open to all interested participants, there is always a risk that larger players will dominate the evolution of the interoperability systems, as in the cse of the industry-led Data Transfer Project. Even if there is broader participation, a collaborative approach is essential once an interface has multiple adopters, otherwise existing implementations are at risk of becoming technical orphans.

Governance is particularly complicated in the sharing of data for pooling, such as in recent EU proposals,[79] or in new institutional forms such as data trusts, which are being piloted in the UK. These aim to provide assurances to individuals and external accountability, while enabling collaboration among organizations, but they have been criticized for not providing enough competition within the market.[80] There are other proposals for collective forms of data governance, for example around a commons model, but these tend to be more relevant when the public sector has a role.[81]

---

74 Stigler Committee on Digital Platforms (2019), Final Report (p.110), https://research.chicagobooth.edu/stigler/media/news/committee-on-digital-platforms-final-report

75 Gasser, U. & Palfrey, J.G (2008), "Breaking Down Digital Barriers: When and How ICT Interoperability Drives Innovation". Berkman Center Research Publication No. 2007-8, https://ssrn.com/abstract=1033226

76 See https://www.cen.eu/work/products/ens/pages/default.aspx#:~:text=A%20European%20Standard%20is%20a,open%20and%20consensus%20based%20process

77 Crémer, de Montjoye & Schweitzer (2019), "Competition Policy for the digital era", European Commission, https://ec.europa.eu/competition/publications/reports/kd0419345enn.pdf

78 Shah, R. & Kesan, J.P. (2008), "Lost in Translation: Interoperability Issues for Open Standards, U Illinois Law & Economics Research Paper No. LE08-026, https://ssrn.com/abstract=1201708

79 See https://ec.europa.eu/info/sites/info/files/communication-european-strategy-data-19feb2020_en.pdf

80 See https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/785547/unlocking_digital_competition_furman_review_web.pdf

81 See https://unctad.org/en/PublicationsLibrary/der2019_en.pdf

Other policy proposals include creating new authorities dedicated to regulating the digital sector. A UK report on digital markets[82] called for the creation of a Digital Markets Unit with extensive new responsibilities, to which the government has agreed.[83] Following the UK's lead, a prominent US report also calls for the creation of a Digital Authority.[84]

Creating a new body that takes some responsibility away from industry self-regulation could iron out difficulties at the governance level, but existing examples of such institutions indicate several possible problem areas:

- the ability of the authority to deal with more technical issues (in terms of skills and resources)
- resolving any overlap/conflict of responsibilities with existing authorities and standards bodies
- preventing "capture" by exactly those dominant players the authority is supposed to help control.[85]

The question further extends to the relationship between frameworks that govern access to private data and those that govern access to public data. The US has a principle of openness for federally funded documents, from weather predictions to NASA imagery. The EU has open data policies enshrined in the regulatory framework for Public Sector Information[86] (including from mapping and weather data) and strong interoperability requirements for public sector software, although these are not always followed. EU data strategy wants to go further and introduce a new Open Data Directive to extend some openness to the private sector. The relation between new open interfaces on the one hand, and separate regimes for public sector bodies on the other, needs to be examined in more detail.

Conflicting objectives and legal clashes

Potential conflicts with other policy objectives and legal regimes are one of the more complex constraints for mandating open interfaces, particularly as many of the services that are the potential subjects of the mandates operate globally across jurisdictions. Here we give a snapshot of the breadth of considerations that should be included in a holistic analysis of mandated interfaces.

Competition law clashes

Competition principles embedded in the highest levels of EU laws allow mandating dominant market players to provide access to data.[87] Recent competition policy proposals considered by the EU look at expanding mandatory interoperability on dominant firms or digital sectors, and lowering the bar for intervention below the current requirements for consumer harm or market domination.[88]

Lowering the thresholds for intervention on competition could create conflicts across jurisdictions, e.g. between the antitrust approaches of the EU and the US. This criticism has already been laid at the portability provisions in the GDPR, which apply to any organization independent of market power or exclusionary practices.[89]

Intellectual property (IP) issues

Interoperability has created some of the most complex legal disputes around the intellectual

82 See https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/785547/unlocking_digital_competition_furman_review_web.pdf

83 See https://www.mondaq.com/uk/antitrust-eu-competition-/824510/uk-to-create-a-39digital-markets-unit39

84 See https://research.chicagobooth.edu/-/media/research/stigler/pdfs/market-structure-report.pdf

85 Ibid.

86 See https://ec.europa.eu/digital-single-market/en/open-data#:~:text='Open'%20public%20data%20are%20PSI,produce%2C%20collect%20or%20pay%20for

87 Article 102, Treaty on the Functioning of the European Union , https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:12008E102:EN:HTML

88 Crémer, de Montjoye & Schweitzer (2019), "Competition Policy for the digital era", European Commission, https://ec.europa.eu/competition/publications/reports/kd0419345enn.pdf

89 Swire, P. & Lagos, Y. (2013), "Why the Right to Data Portability Likely Reduces Consumer Welfare: Antitrust and Privacy Critique", Ohio State Public Law Working Paper 204, https://ssrn.com/abstract=2159157

property of software. Historically, software interfaces were not treated as Intellectual Property (IP). In the US, copyright was rejected and since then there have been attempts to use software patents to control APIs. The high-profile case between Google and Oracle over the Java API has reached the US Supreme Court, for example, and its verdict will have far-reaching implications for interoperability everywhere.[90]

In Europe companies cannot patent APIs, nor is copyright allowed because APIs are defined as "functional characteristics" and not creative works,[91] meaning reverse engineering is allowed.

Other issues concern remediation in cases of disputed intellectual property rights, including instances where a FRAND agreement is in place. For example, a joint policy statement by the US Department of Justice (DOJ) and the US Patent & Trademark Office (PTO) in 2013 stated that in most circumstances an injunction or exclusion order to enforce a standard-essential patent (SEP) covered by a FRAND commitment "may be inconsistent with the public interest". The rationale was that the SEP-holder could use the threat of an injunction in a patent "hold up", in which the patent holder could seek to exclude a competitor or demand higher licensing fees than would have been possible before the patent was included in the standard[92].

However, a more recent policy statement by the DOJ, PTO and the National Institute of Standards and Technology (NIST) has retracted this position, instead asserting that "all remedies available under national law, including injunctive relief and adequate damages, should be available for infringement of standards-essential patents subject to a [FRAND] commitment, if the facts of a given case warrant them."[93]

Data pools and data flows

Interoperability through APIs will enable new data flows, but policymakers might want those flows to be limited to their own jurisdiction. This is particularly, but not exclusively, the case with personal data, due to the regulatory constraints on international data transfers.

For instance, in its Data Strategy the EU is considering creating large-volume data pools for shared access across European industries, enabled by mandatory open interfaces and other policy mechanisms. These pools are meant to accelerate the development of a data-intensive European technology sector, but they will also become attractive for global access.

Restricting access to promote European industry could clash with the proposals to facilitate global data flows that are being discussed at the World Trade Organization and included in many bilateral trade deals. Furthermore, some trade agreements include the forced use of technical standards as a localization barrier, which could affect mandatory interoperability measures.

Mandatory open interfaces could also create additional challenges for understanding what jurisdiction applies, and the enforcement of any national policies across borders. Once data has been transferred across several systems it can be hard to determine jurisdiction and responsibility.

There are already competing dynamics for more extensive cross-border enforcement and isolation. The US CLOUD Act,[94] for example, sets out that US courts can demand data from US companies wherever in the world it is held. The CLOUD Act also allows for direct requests for data by foreign law enforcement if there is a special two-way treaty. The UK and the US have signed the first of those treaties, but the EU and some countries are in discussions. Open interfaces could extend the risks of jurisdictional overreach and increase the complexity of adhering to national laws.

90 Kemp, C. (2020), "Google v Oracle: The Copyright Case of the Decade", Kemp IT Law, https://www.lexology.com/library/detail.aspx?g=ecf5cd6d-2b66-4240-b5d9-efab3c581830

91 See https://curia.europa.eu/jcms/upload/docs/application/pdf/2012-05/cp120053en.pdf

92 U.S. Dept. of Justice and U.S. Pat. & Trade Off (2013), "Policy Statement on Remedies for Standards-Essential Patents Subject to Voluntary F/RAND Commitments", https://www.justice.gov/atr/page/file/1118381/download

93 For a helpful analysis of the updated policy, please see: Levitas, P.J. Kuester Pfaffenroth, S. & Tabas, M. (2020), "DOJ Issues Joint Statement With PTO and NIST on FRAND Injunctions", https://www.arnoldporter.com/en/perspectives/publications/2020/01/doj-issues-joint-statement-with-pto

94 Artzt, M. & Delacruz, W. (2019), "How to comply with both the GDPR and the CLOUD Act" https://iapp.org/news/a/questions-to-ask-for-compliance-with-the-eu-gdpr-and-the-u-s-cloud-act

The challenge of conflicting policies and regulations in a global market: case study of the UK covid tracker

In 2020 the UK Government initiated development of a COVID "track and trace" application. As a case study, this highlights the way in which factors such as market conditions and vendor constraints can affect a government's ability to mandate a technical solution.

The goals of the contact-tracker app are as follows: it runs in a mobile phone and records identifying information for every phone that comes close enough (and for long enough) to result in a statistically significant risk of infection; if the owner of the phone running the app later finds they have COVID-19, the owners of the tracked phones can then be informed.[95]

To have an impact on the pandemic, such an app needs to be used by a significant percentage of people in circulation.[96] It needs a single "clearing house" for notifications. Ideally, it needs to be usable in multiple countries, and it should cater for people with phones/SIMs issued in other countries. This results in several interoperability requirements:

- the app must be compatible with the majority of mobile phones;
- different vendors' phones running the app must be able to detect each other;
- any phone running the app must be able to talk to the server-side component of the system;
- servers must be able to talk to each other, including across borders.

Early on in the pandemic, the UK Government opted for a centralized app similar to one released by the government of Singapore[97] and based on a centralized database of contacts. This design followed the pattern of the Fluphone, a 2011 project by the University of Cambridge[98]. The centralized approach was said to allow more thorough data analysis but was criticized for unlocking the ability to trace exposed citizens without the cooperation of the data subjects. This raised concerns about possible "mission-creep" by law enforcement,[99] by research,[100] and potentially by commercial data users.[101] Notably, the UK Government either did not conduct a Data Protection Impact Assessment before deploying the app, or conducted one but did not share it with the national data protection authority when seeking its advice.

When a prototype was tested in a limited geographic region (the Isle of Wight[102]), it was unable to access the necessary data relating to encounters with other devices, to such an extent that it could never achieve the critical mass needed for it to be effective. On iOS it failed in 96% of cases, and on Android it failed in 25% of cases. This was because those operating systems restrict apps from collecting network data (specifically Bluetooth, in this case) unless the app is active. This is partly for energy-saving reasons, but also because both Google and Apple acknowledge the privacy implications of "always on" network data-gathering. The Government kept the app's source code confidential rather than making it open source, and therefore excluded the expert advice of the UK technical community about how to address such issues.

Earlier engagement with the vendors (Google and Apple), technical experts, and data protection specialists would have highlighted such issues, and either identified solutions, or prompted a change of strategy. In mandating a centralized technical solution, the Government created a specification that made interoperability (at the necessary levels listed above) virtually impossible to achieve - particularly if privacy concerns were to be taken into account. Anecdotally, users were put off by the fact that the app would simply fail to work unless it was running in the foreground at all times. The mandated "always on" approach, which resulted from not having engaged with the manufacturers about what was technically viable, eroded user trust and convenience, and was likely why the app did not achieve the critical mass of adoption needed for it to be effective.

In the meantime, the vendors themselves were researching and implementing an alternative approach. An unusual direct collaboration between competitors, this demonstrated that it was feasible to create a tracker app that was interoperable, decentralized, and privacy-respecting. Started shortly after the UK development, Google[103] and Apple[104] created a design that was implemented as a platform API on both platforms (the Exposure Notification System), with a corresponding, privacy-preserving encryption scheme. Developers in multiple countries were then able to create applications based on the API functions. Many of the apps were open source, including the one created for the German government by SAP[105]—which became the basis for other countries' solutions.[106]

The Google/Apple API protects private information (notably the information of passers-by) while still allowing citizens to opt-in to notification of possible exposure to the virus.[107] The system allows for national supervision of tracing, but preserves the anonymity of the citizens involved until they choose to be identified. The same approach could be used in other contexts where it is important to manage social contact, and where a mobile device would help, but where privacy risks are also a concern (for instance, social distancing or restrictions to groups of a certain size).

95 For an illustration, see https://ichef.bbci.co.uk/news/624/cpsprodpb/B354/production/_112180954_apps_contact_tracingv4_640-nc-2x-nc.png
96 See https://www.independent.co.uk/news/uk/politics/coronavirus-app-uk-nhs-contact-tracing-phone-smartphone-a9484551.html
97 See https://www.bbc.co.uk/news/world-asia-51866102
98 See https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/fluphone2/ - a researcher from Fluphone also advised the NHSX team working on the COVID tracker.
99 Later demonstrated in other countries, https://www.bbc.co.uk/news/world-middle-east-53052395
100 Concerns justified by insider comments: https://www.thebureauinvestigates.com/stories/2020-06-13/where-is-matt-hancocks-contact-tracing-app
101 See https://tech.newstatesman.com/coronavirus/palantir-45-engineers-to-nhs-covid-19-datastore
102 See https://covid19.nhs.uk/isle-of-wight.html
103 See https://www.google.com/covid19/exposurenotifications/
104 See https://www.apple.com/covid19/contacttracing
105 See https://www.bloomberg.com/news/articles/2020-06-16/germany-rejects-criticism-as-covid-tracing-app-finally-goes-live
106 See https://github.com/corona-warn-app
107 See https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-FAQv1.1.pdf

Safeguarding security and privacy

Creating an open interface entails important security and privacy considerations and risks, which should be factored in when looking at public policy and assessing the overall desirability of mandating open interfaces. As outlined below, some of these risks are associated with implementing the interface, while others are related to systemic challenges that might emerge.

Fundamental privacy issues

Many practical discussions around privacy and interoperability focus on particular sectors, such as healthcare data, and the ability to facilitate data portability across providers.

Current discussions are generally framed around the use of APIs, as transferring data in these circumstances is harder to control than if someone is sent an extract of a database, or has to make a bureaucratic request. The aim of APIs is to facilitate data transfers and interaction, but many practical privacy problems that could arise—for instance, if the data is transmitted to the wrong receiver, if data is wrongly shared, or if the data rightfully shared is misused by a legitimate receiver.

How is compliance assured?

In order to safeguard security and privacy it is also important to consider non-technical measures, such as policies and training. Companies such as Facebook and Google have introduced more stringent controls over third-party access[108] and increased user controls to limit the data shared through their APIs. If these measures imply a transfer of control to users, that transfer must be made evident to the user, particularly if it also means the user now bears increased responsibility, and even liability, for what happens to data about them.

End users, or their representatives, therefore need to understand these systems so they can hold companies to account and drive compliance. Data rights are a central plank of privacy protections, starting with the rights to information and access to data. However, the right of access is not the same as data portability. In addition, end users may not be able to deal with the complex data ecosystems that are created through interoperability. Also, as it's difficult for people to know who may be breaching their rights, consumer privacy and civil society organizations should be able to intervene without being instructed to do so by affected individuals.[109]

One way to summarise this is in terms of agency. If users are to have agency in this system, it is not enough for the system to include user tools, settings and options, if users are not also given the information they would need to exercise them. What's more (as some cookie and tracker "consent" panels illustrate, it is possible to design and deploy apparently user-empowering options in a way that is almost guaranteed to discourage the user from doing any such thing.

Data protection and accountability

In order to provide adequate data protection, it is imperative to establish responsibilities. In the EU this means working out who are the "controllers" and "processors" of shared data—according to specific legal definitions of those words.

Data transfers should be underpinned by robust agreements that limit what can be done with the data, retention periods, etc., and that clearly set out proper sharing agreements (for example, in some cases, data pooling will mean joint legal responsibility for data protection). In the EU framework, the "controller" who has the original relation with the subject is primarily responsible, but the GDPR also creates responsibilities for "processors".

The responsibility for privacy also extends to designing and maintaining standards, as this could be another potential point of failure for privacy. The UK government agency GCHQ promotes a standard with a protocol for encrypted protocols that includes a backdoor for access,[110] for

108 See https://apifriends.com/api-security/open-api-impacted-by-data-privacy-data-breaches

109 See https://www.openrightsgroup.org/publications/collective-redress-cheatsheet

110 Murdoch, SJ; (2016) "Insecure by Design: Protocols for Encrypted Phone Calls". Computer, https://discovery.ucl.ac.uk/id/eprint/1476827/

example, and the NSA has been caught weakening NIST cryptography standards.[111] Mandatory systems can provide a single point of failure and require extra vigilance.

Reliability and resilience of systems

Mandatory open interfaces would increase the interconnection among various technical systems. This will increase their interdependence, particularly as more downstream systems become increasingly reliant on upstream systems. The woes of companies dependent on Facebook's platform are well documented. A similar situation of reliance and dependency occurred with Twitter. For many years, it had an ecosystem of applications where users could access the service in a variety of forms. When Twitter closed their API access, those third-party clients disappeared.[112]

Mandatory access could, in principle, provide some certainty around the reliance problem, but it would still create a resilience issue if the service cannot provide continuity for any reason. It could have the perverse effect of centralizing a vertical ecosystem's dependency around a single point of failure.

One problem is that widespread standardization, even to the point of approaching or becoming a monoculture, increases the risk of accelerating the spread of existing vulnerabilities. For example, the OpenSSL toolkit for providing web encryption became so popular that at some point it was running on an estimated 66% of the Internet. When a critical flaw called Heartbleed was discovered, it affected millions of websites.[113] In this case part of the problem was the lack of funding for an open source project,[114] and similar questions about the support for developers could arise in any new framework.

Security responsibilities of whom?

Multi-homing and the emergence of services such as Open Banking can make it hard for users to keep track of who has been authorized to access their data. Some platforms may provide users with this information, but—for example—the Open Banking protocols make it very easy to design apps that connect users to their banks without any central place to keep track, which could be a security risk in itself. Even if the user then deletes the app, it may not be clear whether access to the data at their various banks is still open.

Interoperability could mean that third parties might not be able to fix flaws in a system on which other systems have come to rely. It's not clear what organizations should do at that point, and if the problem affects end users, they won't have a single point of contact to fix it.[115] API portals and other intermediary actors could have an important role to play in providing some assurance.

Questions about responsibility extend to how breaches are handled. In some sectors this may be highly regulated through cyber-security legislation. In serious cases end users should be notified, though in a truly open framework, there could be confusion about which data controller would be responsible for that notification. However, interoperability and data portability could also have some security benefits for users by providing increased data redundancy and account recovery—for example, in services that are deprecated.[116]

---

111 Scientific American (2013), "NSA Efforts to Evade Encryption Technology Damaged U.S. Cryptography Standard", https://www.scientificamerican.com/article/nsa-nist-encryption-scandal

112 Gasser, U. (2015), "Interoperability in the Digital Ecosystem", https://ssrn.com/abstract=2639210

113 Ibid.

114 See https://mashable.com/2014/04/14/heartbleed-open-source/?europe=true

115 Gasser, U. (2015), "Interoperability in the Digital Ecosystem", https://ssrn.com/abstract=2639210

116 Data Transfer Project (DTP), https://datatransferproject.dev/dtp-overview.pdf

Summary

In any area of public policy there will always be a gap between the expected and actual outcome of a particular measure. This gap can be closed by identifying and addressing constraints that might otherwise cause the policy to fail, or bring it into conflict with other objectives, or simply make it impossible to implement. In our analysis, the related considerations can be summarized into three main categories: "Feasibility", "Unintended outcomes" and "Dependencies".

Feasibility

The first step would be to establish the requirements and expected outcomes of the mandate, which should include an assessment of technical compatibility and potential legal conflicts within and across jurisdictions. For instance, is an open interface for this class of activity technically or economically feasible, or even technically possible, and what is required from the specification and resulting interface to achieve its goal? Furthermore, do conflicts between the regulations of differing jurisdictions make it impossible to be open in both places? For example, the EU's General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA) and Japan's Act on the Protection of Personal Information all create duties that need to be observed simultaneously if there are users from their jurisdictions. Some degree of flexibility and iterative design may be needed for an interface to conform with multiple jurisdictions while, at the same time, remaining open and comprehensive.

Unintended outcomes

Underpinning the rationale for mandating an open interface are the opportunities for innovation in products and services. These may result in unforeseen positive outcomes, but it's equally important to consider the possibility of negative outcomes. These includes the potential for barriers to market entry, and consolidation of a dominant player's position. As described in this paper, an interface could be used as an anti-competitive tool: appropriate governance mechanisms should therefore be closely considered. Since interfaces will often evolve over time, any related governance regime should also be reviewed and revised accordingly, to ensure it remains suitable and effective. Conversely, poor choices in the mandate could also lead to large differences in the cost, risk and even feasibility of compliance. For instance, an open interface might give rise to processing and storage costs that the provider cannot to recover from the user or offset elsewhere. Finding the appropriate scope of the mandate, and the necessary mechanisms for collaboration to address issues as they emerge, will be a strong factor in its success. We also recommend an impact assessment for the mandate as a way of mitigating the risks outlined in this section.

Dependencies

Those mandating an open interface should also consider the broader implications for the technical and economic ecosystem. Once access is openly provided to both data and to event streams, new uses—especially in combination with other sources—may unexpectedly become important. Furthermore, mandated interfaces may also introduce new dependencies in the technological ecosystem, raising further questions about security, privacy and the practical governance of the interface. In this light, it is important that any mandate, whether to govern the operations of an existing interface or to create a new one, takes into consideration its role as a building block for the operation of other services. Just as the interface could unlock significant opportunities, it may also become a critical infrastructure component for the services involved.

Appendix A— instant messaging's "family tree"

The diagram below shows the diversity that has evolved in the field of instant messaging from originally shared protocols. For each new branch the degree of "sharedness" decreases, making interoperability limited to a smaller subset of shared features. Implementing interoperability between current end nodes from scratch would be likely to impose significant challenges.

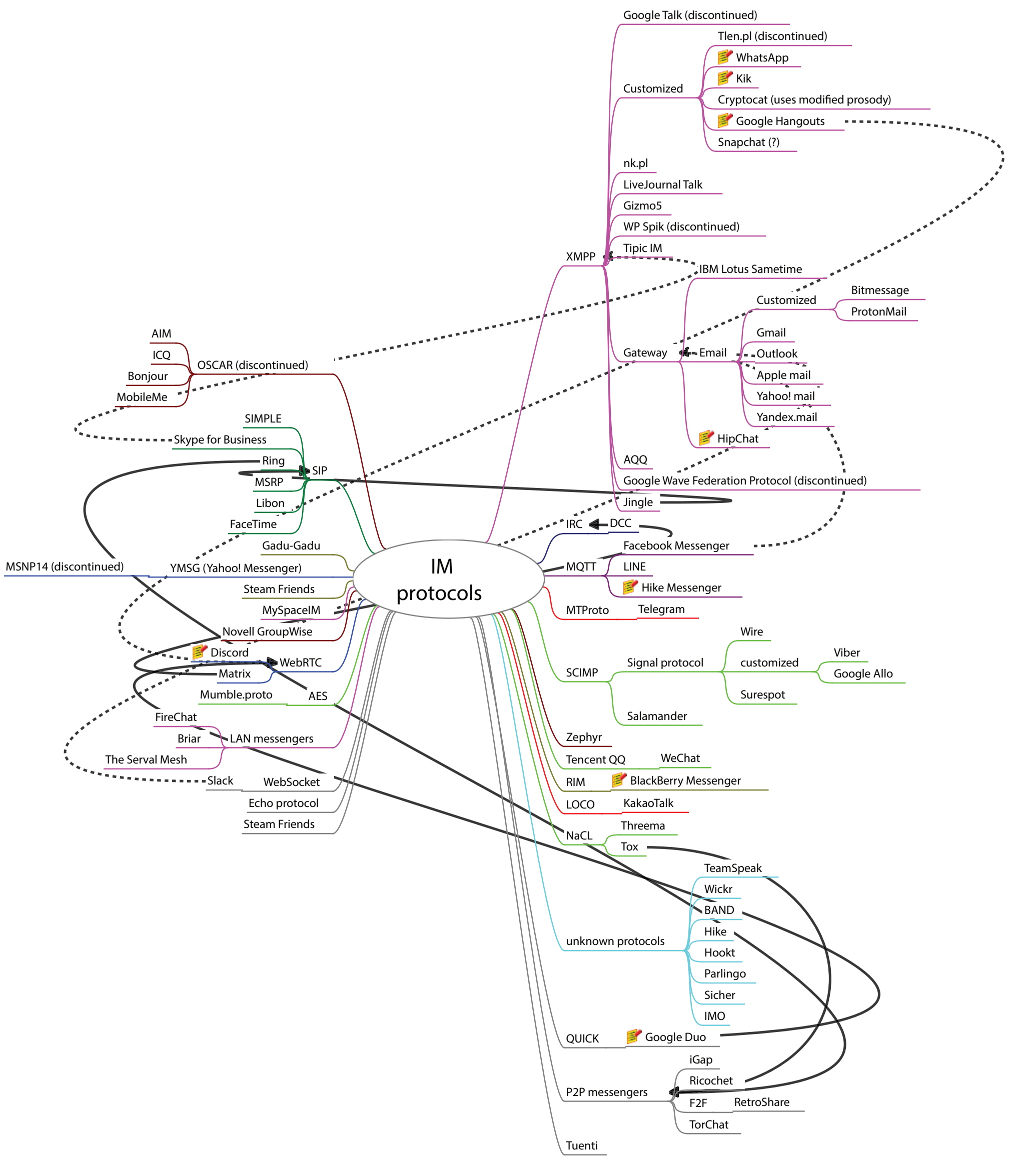

Figure 6.
CC0 image from Wikimedia Commons[117]

117 See https://upload.wikimedia.org/wikipedia/commons/3/30/Instant_messengers_protocols.svg