Data Transfer Project: From Theory to Practice

Brian Willard, Jessie Chavez, Greg Fair, Kyle Levine, Ali Lange, Joyce Dickerson
July 2018

Table of Contents

Abstract

Google's mission is to organize the world's information and make it universally accessible and useful. User trust is critical to achieving our mission, as well as for our business, and we strive to earn and maintain it. Giving users transparency into the collection and use of data, accompanied by powerful and versatile controls, is good for the user and ultimately good for Google. We built our portability tool, "Takeout", to empower users. Data portability provides both transparency and control, creating benefits not just for users and Google, but also for innovation across the broader ecosystem.

Portability helps users try new products or leave a service they don't like, but just providing this functionality is not enough; the quality of the experience is also important. The Data Transfer Project (DTP) extends data portability beyond the ability to download a copy of your data, to providing users with the ability to directly transfer their data into and out of any participating Provider[1]. Many of the design elements of the DTP are based on use cases and preferences that we have observed during our decade of work on portability.

The Data Transfer Project (DTP) is an open source initiative to encourage participation of as many Providers as possible. The goal of the protocols and methodology of the DTP is to enable direct, service-to-service data transfer with streamlined engineering work. The DTP will enhance the data portability ecosystem by reducing the infrastructure burden on both Providers and users, which in turn should increase the number of services offering portability. Ultimately this gives users more options and reassurance that their data won't be 'trapped' in a particular service.

In this paper we describe Google's history of working on data portability, the values that guide our work, the idea behind (and technical foundation of) the Data Transfer Project, and our vision for its future.

Background

Data portability and interoperability are central to innovation. Google has always believed that people should use our products because they provide unique value and features. If a user wants to switch to another product or service because they think it is better, they should be able to do so as easily as possible. This concept of allowing users to choose products and services based on choice, rather than being locked in, helps drive innovation and facilitates competition.

Data portability can also provide security benefits for users. Practical tools that let users backup or archive important information, organize information within multiple accounts, recover from account hijacking, and retrieve data from deprecated services all work to improve user security.

Perhaps most importantly, data portability provides peace of mind to users of our products. Google's technical execution of data portability at scale, and the lessons learned along the way, form the basis of our vision for the future of portability.

[1] See Glossary for defined terms, which are capitalized

Google's Takeout

In 2007, engineers, led by Brian Fitzpatrick in Google's Chicago office, started the "Data Liberation Front."[2] Their goal was "to make it easy for our users to transfer their personal data in and out of Google's services."[3] This statement sounds simple, but a lot of technical thought and consideration went into making it a reality. The early iterations of the Takeout tool focused on adding export features to individual Google products, and later including import features. Engineers focused on making it convenient for users to download data by prioritizing interoperable formats, low cost, and fast transfers.[4]

Four years later, in 2011, Google launched a portability product called "Google Takeout," which created a single location where users could download a copy of their data from a collection of Google products in a variety of industry-standard formats. Allowing users to download their data in multiple formats maximized flexibility, creating many ways for users to use their downloaded data.

Since Google launched Takeout, users have exported more than one exabyte of data and there are currently more than one million exports per month. The scale of Google's operation, both with respect to the number of users and the variety of products, has given Google a unique view into why users want to export their data. Most commonly, users tell us they want to see what data Google has about them or download a copy of their data to keep somewhere else. But they also want to share their data with new services so they can try them out.

"Download Your Data" (formerly known as Takeout) currently facilitates the export of data for a growing list of more than 50 Google products and offers the option to export data to Dropbox and Microsoft OneDrive directly. Google will continue expanding on the functionality offered by providing more export options, like scheduled exports, and additional types of data as new products are developed and new interoperable formats are defined.

As Download Your Data grows, and as more companies create portability offerings of their own, Google continues to be guided by the values and principles that inspired the early engineers of Google's Takeout. From our experience in more than a decade of work in this space, Google has learned a lot about the challenges of moving user data securely in and out of products, including task management, securing credentials, handling failures, and more. We have incorporated these and other lessons into the design and implementation of the Data Transfer Project.

---

[2] https://sites.google.com/a/dataliberation.org/www/home
[3] http://dataliberation.blogspot.com/2009/09/welcome-to-data-liberation-front.html
[4] "There shouldn't be an additional charge to export your data. Beyond that, if it takes you many hours to get your data out, it's almost as bad as not being able to get your data out at all." https://sites.google.com/a/dataliberation.org/www/home

Principles

Google's decision to build a data portability tool began with our own values, but its continued utility relies on principles that apply across the industry. We believe the following principles around interoperability and portability of data promote user choice and encourage responsible product development, maximizing the benefits to users and mitigating the potential drawbacks.

- **Build for users:** Data portability tools should be easy to find, intuitive to use, and readily available for users. They should also be open and interoperable with standard industry formats, where applicable, so that users can easily transfer data between services or download it for their own purposes.

- **Privacy and security:** Providers on each side of the portability transaction should have strong privacy and security measures---such as encryption in transit---to guard against unauthorized access, diversion of data, or other types of fraud. It is important to apply privacy principles such as data minimization and transparency when transferring data between Providers. When users initiate a transfer, they should be told in a clear and concise manner about the types and scope of data being transferred as well as how the data will be used at the destination service. Users should also be advised about the privacy and security practices of the destination service. These measures will help educate users about the data being transferred and how the data will be used at the destination service. More details are in the Security & Privacy section below.

- **Reciprocity:** While portability offers more choice and flexibility for users, it will be important to guard against incentives that are misaligned with user interests. A user's decision to move data to another service should not result in any loss of transparency or control over that data. Individuals should have assurance that data imported to a Provider can likewise be exported again, if they so choose. Ultimately, users should be able to make informed choices about where to store their data. We believe that providing transparency around portability will lead to users preferring Providers that are committed to reciprocal data portability, over those that are not.

- **Focus on a user's data:** Portability efforts should emphasize data and use cases that support the individual user. Focusing on content a user creates, imports, approves for collection, or has control over reduces the friction for users who want to switch among products or services or use their data in novel ways, because the data they export is meaningful to them. Portability should not extend to data that may negatively impact the privacy of other users, or data collected to improve a service, including data generated to improve system performance or train models that may be commercially sensitive or proprietary. This approach encourages companies to continue to support data portability, knowing that their proprietary technologies are not threatened by data portability requirements. For a detailed taxonomy of such data, see ISO/IEC 19944:2017.

- **Respect Everyone:** We live in a collaborative world: people connect and share on social media, they edit docs together, and they comment on videos, pictures and more. Data

portability tools should focus only on providing data that is directly tied to the person requesting the transfer. We think this strikes the right balance between portability, privacy, and the benefits of trying a new service.

The Future: Service-to-Service Portability

The reality of building a product that aligns with these principles has evolved in the decade that Google has been working on data portability. Given our understanding of the current ecosystem, particularly the infrastructure constraints faced by people around the globe, it is clear that the future of portability will need to be more inclusive, flexible, and open. We believe users should be able to seamlessly and securely transfer their data directly from one Provider to another. We also believe the approach outlined below aligns closely with the principles of other thought leaders in the space.[5]

To help make this possible, we've brought our expertise and experience to the Data Transfer Project. Our hope for this project is that it will enable a connection between any two public-facing product interfaces for importing and exporting data directly. This is especially important for users in emerging markets, or on slow or metered connections, as the DTP does not require a user to upload and download the data over what may be low bandwidth connections and at potentially significant personal expense.

Data Transfer Project Overview

The Data Transfer Project is powered by an ecosystem of adapters (Adapters) that convert a range of proprietary formats into a small number of canonical formats (Data Models) useful for transferring data. This allows data transfer between any two Providers using the Provider's existing authorization mechanism, and allows each Provider to maintain control over the security of their service. This also adds to the sustainability of the ecosystem, since companies can attract new customers, or build a user base for new products, by supporting and maintaining the ability to easily import and export a user's data.

Transferring data using canonical formats will not necessarily mitigate problems such as formatting limitations or inconsistent feature support. However, our approach illustrates that a substantial degree of industry-wide data portability can be achieved without dramatic changes to existing products or authorization mechanisms, while still providing a flexible enough platform to adapt and expand to support new formats and use cases brought by future innovation. Additionally, the DTP has been developed to increase participation by motivating Providers to build both export and import functionality into their services.

This paper describes the technical foundation of the Data Transfer Project. Importantly, this work is available in an open-source format to be universally accessible and promote transparency.

[5] http://mesinfos.fing.org/wp-content/uploads/2018/03/PrezDataccess_EN_V1.21.pdf

Use Cases

Individuals have many reasons to transfer data, but we want to highlight a few examples that demonstrate the additional value of service-to-service portability.

- A user discovers a new photo printing service offering beautiful and innovative photo book formats, but their photos are stored in their social media account. With the Data Transfer Project, they could visit a website or app offered by the photo printing service and initiate a transfer directly from their social media platform to the photo book service.

- A user doesn't agree with the privacy policy of their music service. They want to stop using it immediately, but don't want to lose the playlists they have created. Using the DTP open-source software, they could use the export functionality of the original Provider to save a copy of their playlists to the cloud. This enables them to import the playlists to a new music service, or multiple services, once they decide on which Provider they prefer.

- A company is getting requests from customers who would like to import data from a legacy Provider that is going out of business. The legacy Provider has limited options for letting customers move their data. The company writes an Adapter for the legacy Provider's Application Program Interfaces (APIs) that permits users to transfer data to their service, also benefiting other Providers that handle the same data type.

- A user in a low bandwidth area has been working with an architect on drawings and graphics for a new house. At the end of the project, they both want to transfer all the files from a shared storage system to the user's cloud storage drive. They go to the cloud storage Data Transfer Project User Interface (UI) and move hundreds of large files directly, without straining their bandwidth.

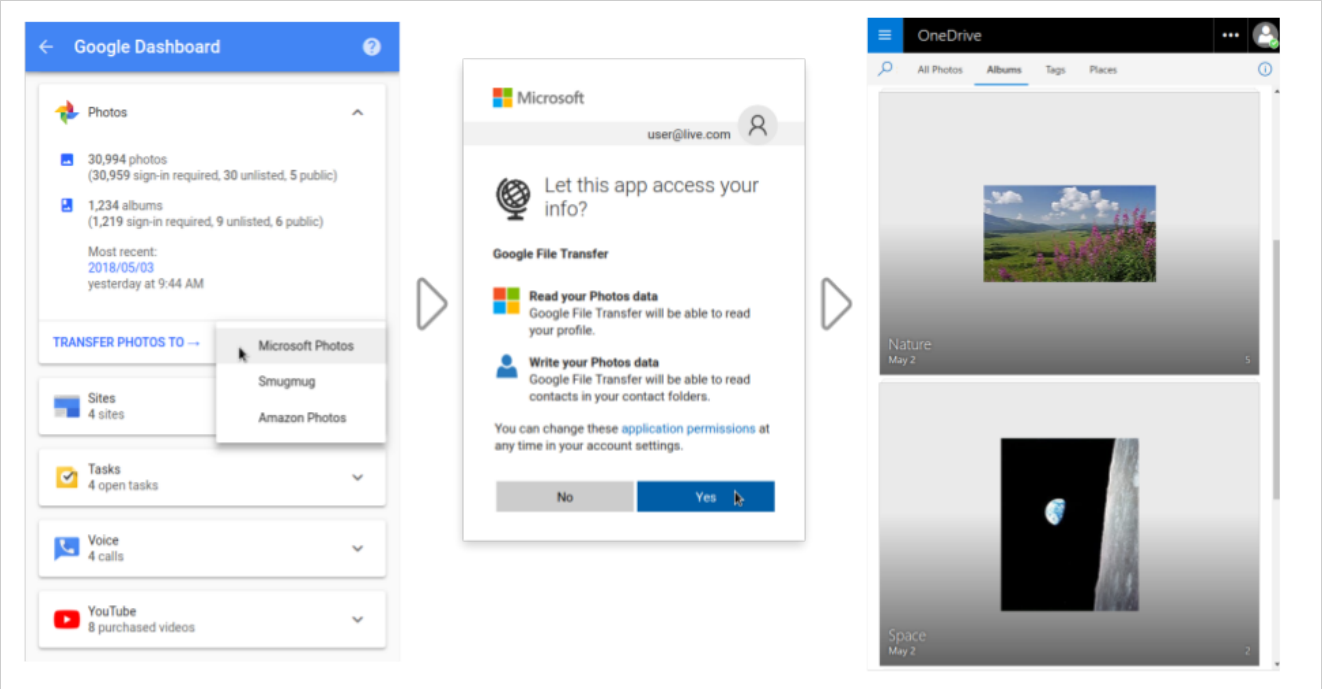
- An industry association for supermarkets wants to allow customers to transfer their purchase history from one member grocer to another, so they can get coupons based on buying habits between stores. The Association would do this by hosting an industry-specific Host Platform of the DTP.

The innovation in each of these examples lies behind the scenes: the Data Transfer Project will make it easy for Providers to meet their users' expectation that they will be able to migrate data with minimal engineering effort. In most cases, the direct data transfer experience will be branded and managed by the receiving Provider, and the customer will authenticate their existing account with both parties. They won't need to see the DTP branding or infrastructure at all as part of the process.

It is worth noting that the Data Transfer Project doesn't include any automated deletion architecture. Once a user has verified that the desired data is migrated, they would have to delete their data from their original Provider using that Provider's deletion tool if they wanted the data deleted.

Generally, any two Providers participating in the DTP ecosystem can enable direct data transfer between them, and participation is self-determined. There is no reason participation would be more attractive to large Providers. In fact, small Providers would reap relatively larger benefits from participating, as the DTP reduces the amount of engineering work needed to build and maintain equivalent functionality.

The illustration below demonstrates what portability enabled by the DTP might look like. In this case, the customer wants to join a new provider (Microsoft) and is requesting their files from their existing provider (Google):



In this hypothetical example, a user wants to move their photos from Google Photos to Microsoft OneDrive. They go to Google's file transfer interface, choose the destination, and hit 'send.' They then must authorize the transfer using both Providers' chosen methods, in this case OAuth. The selected files are automatically copied and routed to the destination, without using either bandwidth or hardware of the user.

Architecture

The DTP solution is designed so that it is easy for a Provider to adopt and enable the standards, and took into consideration the following constraints:

- **Use existing standards; don't create new ones**
  By supporting existing standards where possible (like OAuth and REST), we aim to minimize the foundational work required before the DTP can be built and put into action. Widespread adoption and understanding of existing standards makes this possible. As new standards are developed and adopted, they will be reviewed and, where applicable, incorporated into the DTP.

- **Minimize the work required to add a service**
  We designed the DTP to allow Providers to participate without impacting their own core infrastructure. Providers can build Adapters and enable import and export functionality that works with their existing APIs and authorization mechanisms.

- **Support standard deployment infrastructure**
  The DTP infrastructure was designed with the flexibility to be hosted by anyone, anywhere. See the Deployment for more details.

System Components

As noted above, the DTP system is comprised of three main components:

- **Data Models** are the canonical formats that establish a common understanding of how to transfer data.
- **Adapters** provide a method for converting each Provider's proprietary data and authentication formats into a form that is usable by the system.
- **Task Management Library** provides the plumbing to power the system.

Data Models

The Data Model for transferring files consists of two parts: a file type and the additional metadata needed by the receiving Provider to accurately import the data. For example with photos, the file type might be a standard format such as JPEG, and the metadata would include information such as title, description, album, and so forth.

Data Models are clustered together, typically by industry grouping, to form Verticals. A Provider could have data in one or more Verticals. Verticals could be photos, email, contacts, or music. Each Vertical has its own set of Data Models that enable seamless transfer of the relevant file types. For example, the Music Vertical could have Data Models for music, playlists and videos.

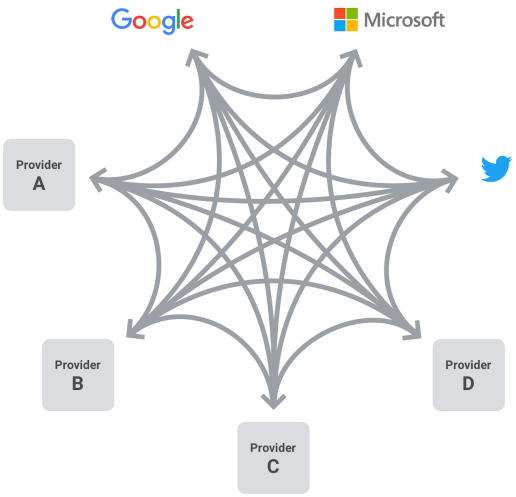
Ideally, a Vertical will have a small number of well-defined and widely-adopted Data Models. In such a situation, the generally accepted standard will be used as the Data Model for that Vertical across Providers. This is not currently the case for most Verticals because Data Models have emerged organically in a largely disconnected ecosystem.

One goal of the DTP is to encourage organizations to use common Data Models in their systems, which will happen if organizations take importing and exporting data into consideration when initially designing their systems or providing updates. Using a common Data Model will significantly reduce the need for companies to maintain and update proprietary APIs.

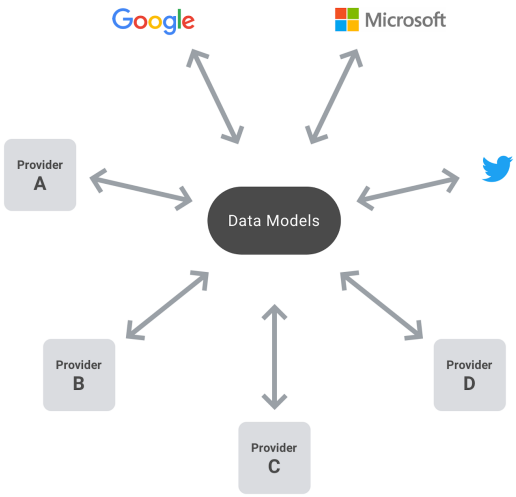
In the case where there is no standard Data Model for a Vertical, companies will want to collaborate and agree upon standardized Data Models, either during DTP development or in

collaboration with external standards bodies. Without collaboration, each Provider could have their own Data Models, and would have to create adapters that would have to support the same number of Data Models as there are companies in the Vertical, which would reduce the usefulness of the DTP.

Even where standard Data Models do exist, collaboration will be an ongoing and mutually beneficial shared commitment as APIs will need to be maintained to handle new features, evolving standards, or innovative new formats.



Without DTP
Each Provider has to build and maintain Adapters for every other Provider's proprietary APIs and, potentially, data formats



With DTP
Each Provider only has to build and maintain an API that supports the DTP Data Models, which are based on standard formats where available

Adapters

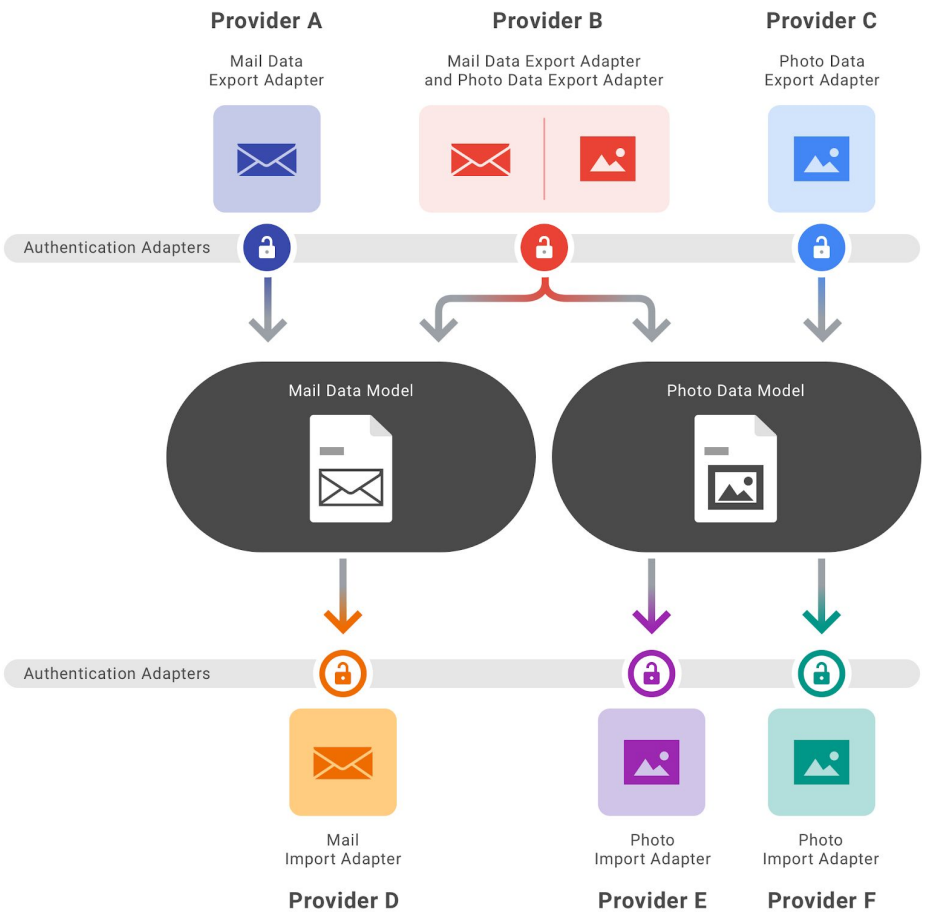
There are two main kinds of Adapters: Data Adapters and Authentication Adapters. These Adapters exist outside of a Provider's core infrastructure and can be written either by the Provider itself, or by third parties that would like to enable data transfer to and from a Provider.

Data Adapters

Data Adapters are pieces of code that translate a given Provider's APIs into Data Models used by the DTP. Data Adapters come in pairs: an exporter that translates from the Provider's API into the Data Model, and an importer that translates from the Data Model into the Provider's API.

Authentication Adapters

Authentication Adapters are pieces of code that allow users to authenticate their accounts before transferring data out of or into another Provider. OAuth is likely to be the choice for most Providers, however the DTP is agnostic to the type of authentication.



Task Management

The rest is just plumbing.

The Task Management Libraries handle background tasks, such as calls between the two relevant Adapters, secure data storage, retry logic, rate limiting, pagination management, failure handling, and individual notifications.

The DTP has developed a collection of Task Management Libraries as a reference implementation for how to utilize the Adapters to transfer data between two Providers. If preferred, Providers can choose to write their own implementation of the Task Management Libraries that utilize the Data Models and Adapters of the DTP.

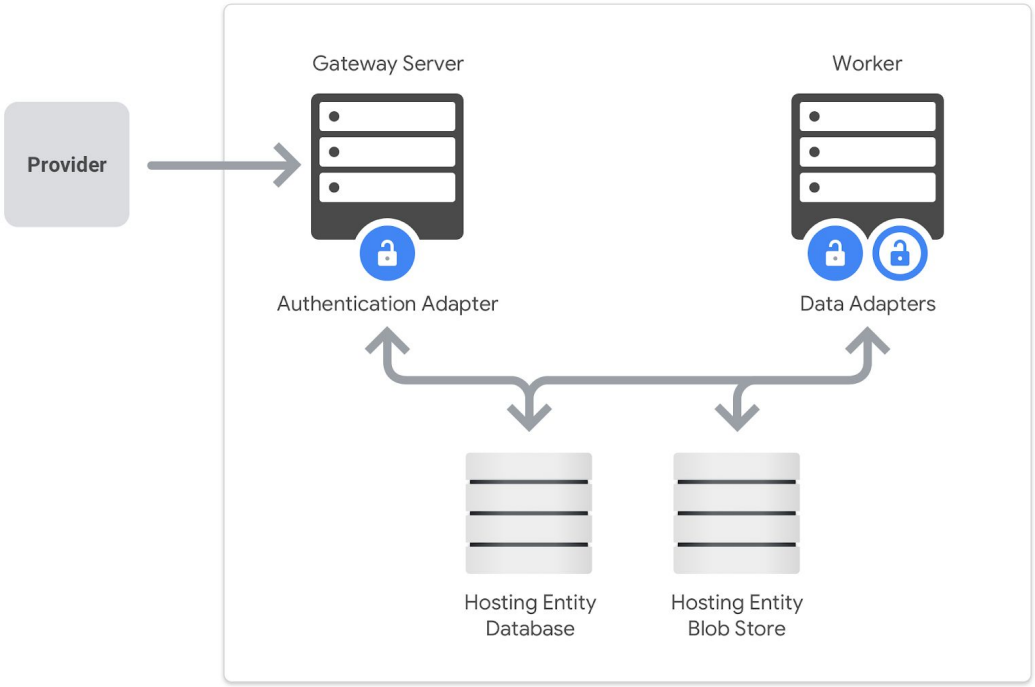


Figure 2: This graphic provides an overview of the interaction between the components of the DTP system. The gateway server facilitates the user's authorization for data export and import via their Authentication Adapter and stores encrypted credential and metadata for the transfer request in the database. A Worker process is assigned to a particular transfer request and invokes the Task Management Library to coordinate and execute export and import tasks, optionally, tentatively storing data in encrypted form in a blob store between export and import.

The Task Management Libraries are built on top of a generic cloud interface so that the Host Platform can be run locally, on a corporate production environment, or on a cloud platform. The cloud interface aims to use only high level abstractions so that it can be implemented on any vendor's cloud platform.

Worker

The Worker utilizes the Task Management Library to execute the Adapters. The Worker is an isolated virtual machine that is created when a data transfer is initiated, and destroyed when that transfer is completed. The Worker generates an ephemeral key when it is created, and that key is destroyed when the Worker is destroyed.

Securely Storing Data

The security of the data passing through the Data Transfer Project is critical. An important objective of the DTP design is that Hosting Entities do not have access to a user's data either in transit or at rest. The DTP security follows industry best practices, and includes requiring transport layer security and storing all data encrypted with the ephemeral key generated by the Worker described above. The DTP system design ensures that administrators do not have access to the encryption key, which protects a user's data.

Details on security measures can be found in the core framework developer documentation on the Data Transfer Project GitHub site (see Resources).

System Access

Each Provider will maintain full control over determining who has access to the data stored on their systems. When a Hosting Entity runs a Host Platform of the DTP, the Hosting Entity will need to request keys from each Provider they would like to be able to transfer data to and from. The DTP will not mediate data access rights between Providers.

This ensures that API quotas continue to be managed by the Provider, thereby helping to mitigate traffic spikes and negative impacts across Providers.

While anyone can contribute Adapters to the DTP, each Hosting Entity decides which Providers they will interact with. Each Hosting Entity determines which Providers to request API keys from, and each Provider chooses which Hosting Entity they grant API keys to. When making these choices, the Hosting Entity should consider the privacy practices of the Provider, as well as its reputation and benefit to the user, in order to ensure the data will be used appropriately. It is up to all actors in the portability ecosystem (Providers, Hosting Entities, Contributors, and Users) to be diligent in ensuring user data is handled safely.

Google's Policy

Google is committed to allowing users to be able to transfer their data to a variety of Providers. However, because both Providers must agree to be connected, each Provider is responsible for protecting the expectations of their users. For example, when determining if a Provider should be included in a Google-controlled User Interface (UI), Google will select Providers subject to the following criteria:

- The Provider meets specified privacy and security guidelines.
- The Provider is providing a legitimate service to the user.
- The Provider has a reliable DTP implementation that doesn't cause unreasonable errors for users, or unreasonable processing requirements for Google.

Deployment

Deploying a Host Platform

Deploying an instance of the DTP involves several steps that allow a Hosting Entity to customize the behavior of the DTP to meet its needs.

Acquire API keys

The Hosting Entity must acquire API keys for all the Providers it wants to be able to transfer data to or from. This allows the Hosting Entity to decide which Providers it wishes to interact with. This step also allows each Provider to have control, if they wish, over which other Providers or Hosting Entities they give keys to. This step may also

require the Hosting Entity to agree to the terms and conditions required by Providers to use their APIs.

Deploy

The DTP is built to be run using docker images. The DTP Open Source Code Repository contains script to deploy the DTP to a number of cloud Providers or locally (see Resources).

Create a User Interface (UI)

When deployed, the DTP exposes a Restful API that allows for HTTPs requests to be sent to it to start and manage transfer jobs. Hosting Entities will likely create a UI on top of the the DTP API to allow their users to easily interact with the DTP.

Deployment Models

There are three models for successfully deploying a Host Platform of DTP: distributed, centralized, and self-managed. Each has advantages and disadvantages and no one solution is right for all use cases.

Distributed

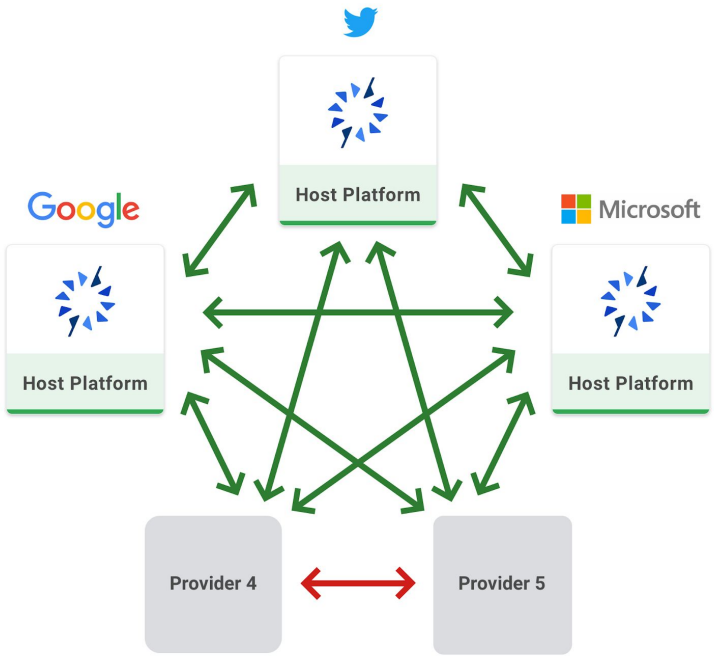
In a distributed hosting model, the Provider becomes the Hosting Entity and runs a Host Platform of DTP. When a user wants to transfer data to or from a Provider, the user initiates the transfer using the Host Platform of the Provider they are initiating the transfer to or from.

> Example: Google and Microsoft are both running Host Platforms of DTP. If a user goes to the Microsoft UI and initiates an import from Google, they would use the DTP Host Platform run by Microsoft.

The advantage of distributed hosting is that data is never handled by a third-party. Only the source Provider and the receiving Provider have access to the data itself. The disadvantage is that it limits transfers to Providers that are running DTP Host Platforms.

Additionally, every Provider having to maintain a separate Host Platform could create excessive Host Platform overhead costs.

Distributed DTP Environment



In this example of a distributed model, Microsoft, Twitter, and Google are each a Hosting Entity, and can transfer data from each other usingDTP Adapters and Data Models. Providers 4 and 5 have DTP Adapters but have chosen not to run a Host Platform. Users can transfer data from Google Microsoft, and/or Twitter to Providers 4 and/or 5. However users can not transfer data directly between Providers 4 and 5 because neither Provider is running a Host Platform of DTP.

Centralized

In a centralized system, an independent third-party is the Hosting Entity, and sets up and configures a DTP Host Platform to handle requests to and from any Provider (or a subset of them).

> Example: An NGO interested in data portability agrees to run a Host Platform of DTP. Users can go to the NGO's portal and transfer data between any two Providers that have Adapters in the DTP.

The advantages of this system are that many small companies will want to be part of the DTP, yet they don't have the resources or expertise to run a Host Platform themselves. With the centralized system, they will only have to write and maintain an Adapter.

Importantly, this requires that a centralized third-party be both trustworthy and highly skilled in data protection. The third-party could alter the DTP code to gain access to a user's data and thus a Provider could be a target of attackers. Additionally, the third-party would have to find a means of financially supporting this initiative, since hardware and bandwidth do have a cost.

Self-managed

In a self-managed environment, a user can download and run a copy of the DTP either locally on their machines or in their private cloud instance.

The advantage of self-managed is that it allows users to fully control the transfer experience. For example, users could transfer their data between Providers with end-to-end encryption and not have to upload or share their private keys. Having this option also ensures that if a Provider stops running a Host Platform, users still have the ability to transfer data to and from that Provider. The disadvantage is that running a DTP Host Platform is more complex and resource intensive than most users will be able to take on.

Security & Privacy

The security and privacy of user data is a foundational principle of the Data Transfer Project. Because there are multiple parties involved in the data transfer (the user, Hosting Entity, Providers, and Contributors) no one person or entity can fully ensure the security and privacy of the entire system. Instead, responsibility is shared among all the participants. Here are some of the responsibilities and leading practices that contribute to the security and privacy of the DTP.

Data minimization[6]

When transferring data between Providers, data minimization should be practiced. Practically, this means that the receiving Provider should only process and retain the minimum set of data for the user that is needed to provide their service. The sending Provider should provide all needed information, but no more.[7]

User notification

The Hosting Entity should configure their Host Platform to notify the user that a data transfer has been initiated by the user. Ideally, the user of the source and destination account are the same. However, user notification is designed to help protect against situations where that is not the case, and so notifications alerting the user of the transfer request should be sent to both the

[6] De Hert, Paul, Malgieri G. 'User-provided personal content' in the EU: digital currency between data protection and intellectual property. International Review of Law, Computers and Technology, Volume 32, 2018. https://doi.org/10.1016/j.clsr.2017.10.003

[7] DTP won't delete data from the sending Provider as part of the transfer. However, participating Providers should allow users to delete their data after a successful transfer has been verified.

source account and the destination account. Depending on the sensitivity of the data being transferred, the Hosting Entity should consider delaying the start of the transfer so that the user has the opportunity to cancel the transfer after receiving the notification.

Rate Limiting

Hosting Entities, as well as Providers, should consider rate limiting the number and frequency of transfers for a given user. This approach can help limit the impact of an account compromise. The trade off between ease of use and security with this method means there is not a one size fits all answer as to what rate limit should be set. Instead, Providers and Hosting Entities should evaluate the sensitivity of the data, as well as known and possible attacks, when determining the appropriate rate limiting.

Token Revocation

When a transfer is completed, the DTP will attempt to revoke the authorization tokens used for the transfer. Providers should ensure their API supports token revocation. This approach ensures that if one of the security mechanisms is compromised, a second layer is in place to provide protection (defense in depth) to ensure that if a token is leaked, its effectiveness will be limited to the duration of the transfer.

Minimal Scopes for Auth Tokens

Providers should offer granular scopes for their authentication tokens. This provides two benefits: first, providing transparency into exactly what data will be moved; second, as a defense in depth mechanism so that if tokens are somehow leaked they have the minimal possible privilege. At a minimum there should be read-only versions of all the scopes so no write/modify/delete access is granted on the sending Provider.

Data retention

The DTP stores data only for the duration of the transfer job. Also, all data passing through the system is encrypted both at rest and in transit. Specifically, all data stored at rest is encrypted with a per-user session key that is created and stored in memory of the ephemeral virtual machine that is used to process a single user's job. The Hosting Entity and Provider are responsible for ensuring that any stored aggregated statistics maintain user privacy.

Abuse

Providers and the Hosting Entity (if separate from the Provider) should have strong abuse protections built into their APIs. Due to the fact that the DTP retains no user data beyond the life of a single transfer, and that there might be multiple Hosting Entities utilized in an attack, the Providers have the best tools to be able to detect and respond to abusive behavior. Providers should carefully control which Hosting Entities are able to obtain API keys. Providers are also encouraged to have strong security around granting auth tokens. Examples of this include requiring a reauthentication or asking security challenge questions before granting access.

Shared Responsibilities Table: Security and Privacy

Task	User	Provider-exporter	Provider-Importer	Hosting Entity	DTP System
Data Minimization	Selects data to transfer	Provides granular controls of what data to export	Discards any data not needed for their service	Configure only appropriate transfer Providers	N/A
Rate Limiting	N/A	Implements	N/A	Sets reasonable limits to prevent abuse	Supports Provider specific rate limiting
User Notification	Receives and reviews notification of transfer	N/A	N/A	Configure mail sender and delay policy	Send notification, optionally with delay to allow for cancellation
Token Revocation	May need to manually revoke tokens if Provider doesn't support automated revocation	Support Token Revocation	Support Token Revocation	N/A	Revoke Auth tokens after use (if supported by Providers)
Minimal Scopes for Auth Tokens	Verify Appropriate Scopes requested	Implements granular scopes	Implements granular scopes	N/A	Requests minimal scopes for each transfer
Data Retention	Transfer of data is not deletion, user should delete source data if desired	Store only data needed to prevent fraud and abuse	Only retain imported data in compliance with privacy policies; Store metadata needed to prevent fraud and abuse	Configures system to not retain any identifiable information	Retains no data after transfer completed
Abuse	Protect account credentials (strong passwords, two-factor authentication, etc.)	Implement appropriate fraud and abuse protections on APIs	Implement appropriate fraud and abuse protections on APIs	Implement appropriate fraud and abuse protections on UI	Encrypts data in transit and at rest using ephemeral key; Uses isolated/dedicated VMs per transfer

Ecosystem Issues

Inconsistent API Landscape

Despite the project's emphasis on using open web and standard technologies, there still exist technical and public policy issues. One obstacle the DTP will face is that the API landscape is inconsistent, particularly with respect to the flexibility afforded by a service. The problem is twofold: some companies lack open APIs and others don't support or maintain them sufficiently to enable service-to-service portability at scale. Even when companies offer different types of open APIs, they may have restrictions in their Terms of Service prohibiting use cases like the DTP from using them. By restricting how users can relocate their data, these rules discourage consumers from trying new services.

Reciprocity

A healthy data portability ecosystem requires Providers that allow equivalent import and export functionality. Providers that import data but don't allow a similar level of export pose a risk to users by trapping their data into a service. There are several possible ways to promote reciprocity in the DTP ecosystem. We have identified several methods, which we list below, and will work with Providers to further explore these and other options.

In source code

Contributions into the main Source Code Repository, hosted on GitHub, are encouraged to contain an exporter coupled with each importer. This is to ensure at least an attempt at reciprocity.

Transparency

Each Hosting Entity is encouraged to provide aggregated statistics about problems users encountered when importing and exporting data to Providers. This aims to ensure that Providers are maintaining their exporting functionality to a similar level of reliability as their importing functionality.

Automated fidelity test

Hosting Entities can establish test accounts with Providers and do periodic imports and exports of data to each Provider to ensure that data is exported with the appropriate fidelity. This information can again be provided in a transparent way to users at import time to ensure users can make an informed decision when choosing which Providers to entrust with their data.

Data Portability Provider Pledge

Providers can work together to create a data portability pledge that requires them to follow best practices on portability. Hosting Entities can seek to support Providers that

commit to the pledge, user interfaces can display Providers that commit to the pledge, and reports can be published on the state of the ecosystem with regards to reciprocity.

Personal Information Management Systems (PIMS)

Personal Information Management Systems have some overlapping sets of features with the DTP. Both allow users to control their data and send it to different online Providers.

For data minimization, we believe the DTP is a better alternative than PIMS since the DTP avoids storing the same data in multiple places. Typically, PIMS aggregate data from connected Providers and create an extra copy of a user's data appended with data from multiple sources. Using the DTP, data resides in the originating Provider and destination Provider without a new copy being created at a third-party. The data only exists at the Providers a user chooses.

PIMS are also subject to the risks and disadvantages outlined above in the description of the Centralized Deployment Model.

The Data Transfer Project makes it easier for PIMS to interact with Providers. Without the DTP, each PIMS would have to create Adapters for each data type and Vertical for each Provider, thereby greatly increasing the number of Adapters they have to build and maintain.

Implementation

The success of the Data Transfer Project relies on generating an interested community of participants, including Providers and users who want to transfer their data. The implementation described in this paper currently resides in its prototype stage on GitHub, where we invite others to join the project: www.datatransferproject.dev. Links to specific repositories and guides are listed in the Resources section at the end of the paper.

The Open Source Code Repository for the Data Transfer project is also available on the Data Transfer Project GitHub site listed above. The Repository contains a prototype implementation that supports data transfers for a handful of initial Verticals (photos, mail, contacts, etc.), Data Models, Authentication Adapters, and Data Adapters, as well as components needed to run the DTP infrastructure on two cloud Host Platforms (Google Cloud Platform and Microsoft's Azure).

The GitHub repository also contains developer documentation on how to add services to the DTP, how to integrate an existing Adapter, and how to deploy a Host Platform either locally or on a server.

Our goal is for the developer community is to extend the DTP Platform to support more Verticals, Providers, and Host Platform options.

Ways to Participate

Add a Provider

New Providers can join the DTP using the set of interfaces described in the *Provider Integration Guide* on GitHub. It requires writing a Data Adapter, Auth Adapter and, potentially, a new Data Model. See the *Provider Integration Guide* on GitHub for more details (see Resources)

Contribute to the Core Framework

Core contributions are welcome, including support for additional Providers, Host Platforms, Data Models, storage systems, etc. See the *Developers Guide* on GitHub (see Resources).

Conclusion

The Data Transfer Project is something Google is excited about and we encourage the industry to adopt a bolder and broader view of the data portability ecosystem. We plan to continue iterating on our design, fostering thought leadership in portability, and publishing updated information about our work and proposals.

Glossary

- **Adapters -** Adapters exist outside of a Provider's core infrastructure and provide the translation between a Provider's core infrastructure and the DTP environment.
    - Data Adapters translate a Provider's APIs into Data Models, and vice versa
    - Authentication Adapters translate from a Provider's Authentication to the DTP Authentication
- **Contributors** - Contributors are official members of the Data Transfer Project. They are most likely Providers, but may also be organizations who are interested in enabling data transfer among their members. Contributors contribute in many ways to the DTP, including contributing code, tools, advice and insights.
- **Data Model** - Data Model is used to describe the file type and associated metadata to describe elements in a Vertical. For example, in the Photos Vertical, a Data Model would include a file type (.jpg, .tiff) and the associated metadata needed to fully describe the .jpg as a photo, such as title, description and album, if applicable.
- **DTP** - Data Transfer Project
- **Host Platform** - a Host Platform is the technical environment where a DTP instance can be hosted. This can be a cloud environment, enterprise infrastructure, or a local server. As of July 2018, the supported cloud host platforms include Google Cloud Platform and Microsoft Azure.

- **Hosting Entity** - a Hosting Entity is the entity that runs a Host Platform of the DTP. In most cases it will be the Provider sending or receiving the data, but could be a trusted third-party that wants to enable data transfer among a specific group of organizations.
- **Provider** - Providers are any company or entity that holds user data. Providers may or may not be Contributors. Provider is similar to Cloud Service Provider as defined in ISO/IEC 17788:2014 section 3.2.15
- **Task Management Library** - The Task Management Library is a set of system components designed to coordinate and manage tasks that export and import data including retrieving data, pagination when applicable, handling exceptions utilizing retry strategies, re-creation of folder/container structures, and importing of data.
- **User** - Users are any person who interacts with a Provider. Users are interested in being able to manage and move the data they have stored in a Provider's infrastructure.
- **Vertical** - Verticals represent a collection of Data Models that make up a generic category of data. Some Verticals initially supported in the DTP include Photos, Calendar, Tasks, and so on.
- **Worker** - the Worker is an isolated virtual machine created when a data transfer is initiated, and destroyed when it is complete. The Worker holds the security key, and ensures that it does not exist longer than needed for the data transfer to complete.

Resources

- **Data Transfer Project GitHub Site:**
  - **Open Source Code Repository** github.com/google/data-transfer-project
  - **Provider Integration Guide** github.com/google/data-transfer-project/blob/master/Documentation/Integration.md
  - **Developers Guide** github.com/google/data-transfer-project/blob/master/Documentation/Developer.md
- **Mailing list:** dtp-discuss@googlegroups.com

Acknowledgements

Thanks to the following for their feedback and input: John Breyault (National Consumers League), Geoffrey Delcroix, Joe Hall (Center for Democracy & Technology), Babak Jahromi (Microsoft) and Joe Jerome (Center for Democracy & Technology). Thanks to Jeff Ma for the DTP logo and graphics.