IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

Daedalus Blue, LLC, Plaintiff, v. Dropbox, Inc., Defendant.	Civil Action No.: _____ **DEMAND FOR JURY TRIAL**

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff, Daedalus Blue, LLC "(Daedalus") files this Complaint against Defendant Dropbox, Inc. ("Dropbox" or "Defendant"), and in support thereof alleges as follows:

NATURE OF THE ACTION

1. The novel inventions disclosed in the Asserted Patents in this matter were invented by International Business Machines Corporation ("IBM"). IBM is and has been a pioneer in the computing world. Every year, IBM spends billions of dollars on research and development to invent, market, and sell new technology, and IBM obtains patents on many of the novel inventions that come out of that work, including the Asserted Patents. The three patents asserted in this case, U.S. Patent Nos. 7,542,957, 8,176,269, and 8,131,726 (the "Asserted Patents") are the result of the work from IBM researchers to improve the fields of online application security and document storage.

2. Over the years, the inventions claimed in the Asserted Patents have been licensed to many companies, including but not limited to Amazon Web Services and Oracle Corporation.

3. On information and belief, Dropbox was also previously licensed to the Asserted Patents and practiced the inventions claimed pursuant to that license. On information and belief, that license expired and since that time Dropbox's activity has been unlicensed and unlawful.

THE PARTIES

4. Daedalus is the current owner and assignee of the Asserted Patents.

5. Daedalus is a Delaware limited liability company with its principal place of business located at 51 Pondfield Road, Suite 3, Bronxville, NY 10708.

6. On information and belief, Dropbox is a publicly traded corporation organized and existing under the laws of the State of Delaware and is registered to do business in the State of Delaware. On information and belief, Dropbox has a principal place of business at 1800 Owens Street, San Francisco, CA 94158. Dropbox may be served with process through its registered agent, Corporation Service Company, 251 Little Falls Drive, Wilmington, Delaware 19808.

7. Dropbox conducts business in Delaware and in the District of Delaware, as set forth below.

JURISDICTION AND VENUE

8. This is an action arising under the patent laws of the United States, 35 U.S.C. § 101, *et seq*. Accordingly, this Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

9. This Court has personal jurisdiction over Defendant Dropbox because it is incorporated in the State of Delaware and therefore Dropbox resides in this District.

10. The Court has personal jurisdiction over Defendant Dropbox because it markets its products and services through an interactive website located at www.dropbox.com that is publicly accessible to consumers throughout the United States and this District.

11. Venue is proper in this District pursuant to at least § 1400(b), at least because Defendant Dropbox is incorporated in the State of Delaware and therefore resides in this Judicial District pursuant to 28 U.S.C. § 1400(b).

12. Venue is proper in this District pursuant to at least 28 U.S.C. § 1319(b)-(c) because Defendant Dropbox is formed under the laws of Delaware and therefore resides in the District of Delaware which subjects it to the personal jurisdiction of this Court.

FACTUAL ALLEGATIONS

13. The IBM inventions contained in the Asserted Patents in this case relate to groundbreaking improvements to computer functionality and computer security. The techniques IBM developed are described in the Asserted Patents and relate to computer networks and have particular application in the fields of online application security and document storage as will be further described below.

A. U.S. Patent No. 7,542,957

14. On June 2, 2009, the U.S. Patent and Trademark Office duly and lawfully issued U.S. Patent No. 7,542,957 ("the '957 Patent"), titled "Rich Web application input validation." A true and correct copy of the '957 Patent is attached as **Exhibit 1**.

15. Daedalus is the owner and assignee of all right, title, and interest in and to the '957 Patent, including the right to assert all causes of action arising under said patent and the right to any remedies for infringement of it.

16. The '957 Patent claims systems and methods for web application security—specifically a validation engine for validating requests for a web application by utilizing validation logic comprised of a set of validation rules. (*See* Ex. 1 at Abstract).

17. The claims of the ’957 Patent describe claim elements, individually or as an ordered combination, that were non-routine and unconventional at the time of the invention in 2009 and an improvement over prior art, as they provided a way (not previously available) to develop flexible and reusable authentication and validation rules to manage access to rich web applications.

18. Specifically, the ’957 Patent describes, among other things, novel systems and methods that improve the functioning of a computer, including improvements to the way in which data is accessed and retrieved through improved methods for validating access to online systems. The inventive technological improvements described in the ’957 Patent solved then-existing problems in the field of rich web applications. For example, as described in the specification of the ’957 Patent, developers of web applications were required to code and implement validation engines for each application. (Ex. 1 at 3:25-51). This is an issue as described in the specification:

> To overcome this problem, custom code (for example in JavaScript, C++, or Java) may be needed to validate values which cannot be handled by the existing validation engine capabilities. Writing and maintaining custom validation code is not efficient. Since Web application data validation logic is repetitive, the advantage of pre-defined rule types may be lost. Custom validation require a greater level of expertise from the rules writer (knowledge of code programming).

(Ex. 1 at 3:43-51).

19. Fig. 5 of the patent shows a rule that is difficult to implement via preexisting methods:

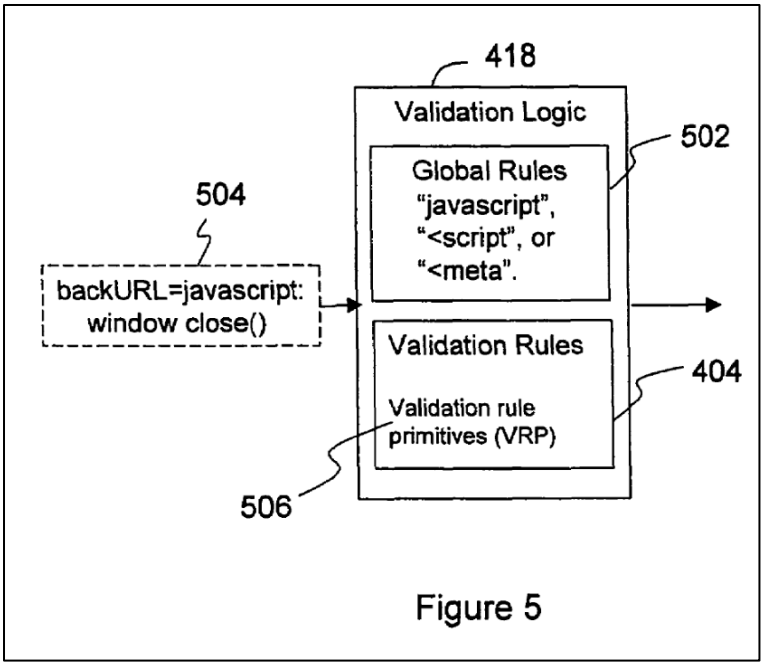


Figure 5

(Ex. 1 at Fig. 5).

20. This shows a request parameter that contains a sub-value that is valid for the parameter, but invalid as part of a global rejection rule. (Ex. 1 at 8:38-41). The global rejection rule rejects any parameter value containing "javascript," "<script," or "<meta." Ex. 1 at 8:41-43. If the global rule is applied, the value in 504 will be rejected because it contains "javascript." (Ex. 1 at 8:45-47). However, if a value containing "javascript" such as in 504 should be allowable, skipping the global rule is not a proper solution to allow this kind of input since other elements of the rule (for example prohibiting "<script") could be used. (Ex. 1 at 8:45-50). The inflexibility of these types of rules makes modifying for different inputs difficult.

21. Moreover, another disadvantage of code driven rules "is that once an application is deployed in an environment, policies will often prevent modifications to the installed code." (*Id*. at 3:52:54).

22. The patent recognizes a need for improved systems and methods:

> Therefore, there is a need for a richer and yet simple to define rules applied by a validation engine. The rules capabilities allow tight validation of complex Web

> application data without the need for customized validation code. There is a need for the rules syntax to be adapted for human handling, either by using human readable rule definitions, or by manipulating a tool. There is a need for the rules syntax to help write, to Verify correctness, to ensure completeness, and to facilitate updates of the rules. There is a need for a prompt fix when a security vulnerability is newly discovered, a rules upgrade is preferable than a code upgrade. The update of validation rules is flexible and quick to implement.

(Ex. 1 at 4:24-35).

23. The ’957 Patent presents a technical solution to these technical problems. The technology described in the patent presents a method for developing powerful, and yet flexible, validation rules primitives or “VRPs” that can be processed as described in Fig. 6:

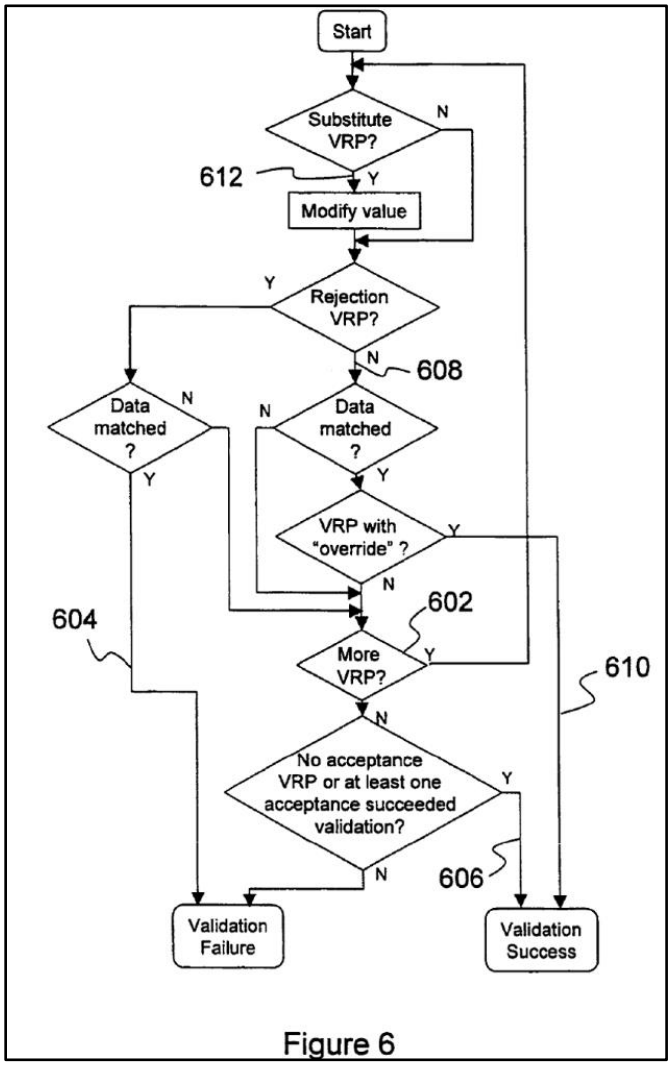


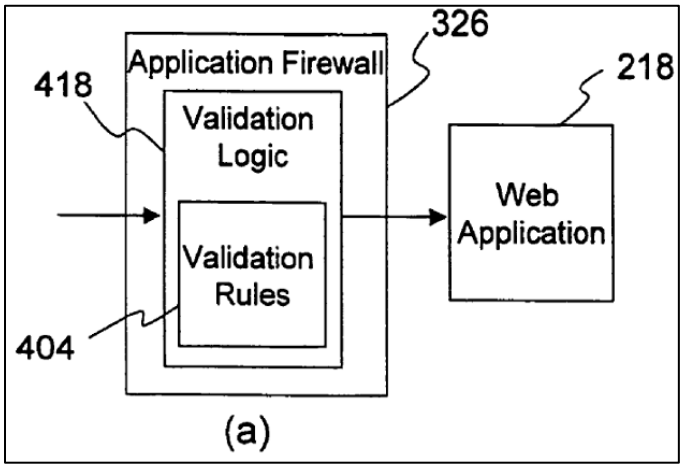
Figure 6

24. One method of how the ’957 Patent solves these problems is in claim 2:

> A method for validating a request to a Web application, the request having a data comprising:
>
> creating a validation engine in a programmable processor, the validation engine comprising a validation logic, said validation logic comprising a validation rule, said validation rule corresponding to a defined plurality of data elements;
>
> loading said validation rule;
>
> applying said validation rule to said data elements;
>
> and sending said request to the Web application.

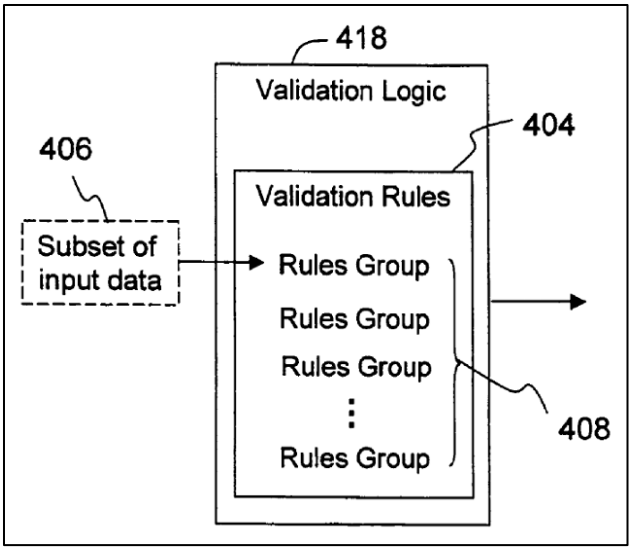
(Ex. 1 at claim 2).

25. The claims introduce a novel method using defined yet easily combined validation rules to create flexible rules that can be implemented quickly and efficiently. In one described embodiment, the patented methods and systems are implemented within an application firewall:



(Ex. 1 at Fig. 4(a); *see also id.* at 8:61-63).

26. The new technology involves validation rules nested under a validation logic that determines whether to provide data provided by one system to a web application. One embodiment of the creation of these rules is set forth in Fig. 4(b):



(Ex. 1 at Fig. 4(b)).

27. In an exemplary system, these validation rules are made up of a series of rules primitives which form the building blocks of a validation system. (Ex. 1 at 10:52-54). Using one or more of these primitives, a rule may specify acceptable or unacceptable values for a particular parameter. (Ex. 1 at 9:5-11). These primitives are processed in the order in which they appear, yet allow for complex combinations such as overrides, substitutes, etc. (Ex. 1 at 10:55-67). By setting up a system for validation logic made up of elemental rules that can be mixed and matched, the patented system solves a problem with requiring rigid validation for each web application.

B. U.S. Patent No. 8,176,269

28. On May 8, 2012, the U.S. Patent and Trademark Office duly and lawfully issued U.S. Patent No. 8,176,269 (the "'269 Patent"), titled "Managing metadata for data blocks used in a deduplication system." A true and correct copy of the '269 Patent is attached as **Exhibit 2**.

29. Daedalus is the owner and assignee of all right, title, and interest in and to the '269 Patent, including the right to assert all causes of action arising under said patent and the right to any remedies for infringement of it.

30. The ’269 Patent describes systems and methods for data deduplication by breaking files into data blocks, maintaining information on whether any of those blocks becomes unreferenced, and adding a data block reference for one of those blocks to another file if the data block matches data in a new file. (Ex. 2 at Abstract).

31. The claims of the ’269 Patent describe elements, individually or as an ordered combination, that were non-routine and unconventional at the time of the invention in 2012 and an improvement over prior art, as they provided a way (not previously available) to manage unreferenced data blocks in a database system using data deduplication on a block basis. For example, when moving or deleting files, it was unconventional to maintain the data blocks that were unneeded while marking them such that they were available to reference at a later time.

32. Specifically, the patent claims improvements over data deduplication systems by allowing reuse of data blocks after they have become unreferenced (e.g. moved or deleted). *See* (Ex. 2 at 6:3-23). The ’269 Patent recites specific technical solutions that are an improvement over previous computerized data management systems. As described in the specification, certain prior art file systems attempted to deduplicate data by splitting data into blocks and using references to stitch those blocks together, thereby enabling the reuse and repurpose of data blocks. (Ex. 2 at 1:23-48). The ’269 Patent is an extension and improvement on this prior art, as it solves the problem of how to more effectively manage file deletion and restoration, common file system processes in non-deduplicating systems, for the new paradigm of block-based deduplication. (Ex. 2 at 1:49-2:4).

33. This is a technical improvement over a uniquely technological problem and the patent claims describe novel storage methodologies that improve the functioning of a block-level reference-based storage system that is not routine or conventional.

34. One such method is described in claim 1:

A method, comprising:

maintaining file metadata for files having data blocks in a computer readable storage device, wherein at least one of the files has file metadata indicating that the file has multiple data blocks;

maintaining data block metadata for each data block in the computer readable storage device, wherein the data block metadata for one data block includes a data block reference and content identifier identifying content of the data block, wherein the file metadata for each file includes the data block reference to each data block in the file;

determining an unreferenced data block in the computer readable storage device that has become unreferenced;

indicating the data block metadata for the determined unreferenced data block as unreferenced data block metadata; and

adding the data block reference of the unreferenced data block metadata in the computer readable storage device to file metadata for an added file that includes multiple data blocks including one data block having content matching the content of the unreferenced data block according to the content identifier in the unreferenced data block metadata.

(Ex. 2 at claim 1).

35. Specifically, the ’269 Patent describes a novel method relating to data blocks that are deleted or moved and therefore become unreferenced. (Ex. 2 at 5:26-37). Rather than deleting the data altogether, the invention of the ’269 Patent retains this data for example for a period of time, or when the system determines that the data is likely to become referenced again in the future. (Ex. 2 at 5:37-6:2). This data then is available to be referenced by new files added to the system even when it had previously been unreferenced. (*See id.* at 4:30-5:20). The patent office recognized the novelty of this invention, as the patent examiner noted that “none of the prior [art] of record teaches or fairly suggests the combination including the limitation of . . . determining an unreferenced data block in the computer readable storage device that has become

unreferenced; indicating the data block metadata for the determined unreferenced data block as unreferenced data block metadata; and adding the data block reference of the unreferenced data block metadata in the computer readable storage device to file metadata for an added file that includes multiple data blocks including one data block having content matching the content of the unreferenced data block according to the content identifier in the unreferenced data block metadata." App. No. 12/165,540, 1/09/2012 Notice of Allowability.

36. Thus the '269 Patent describes an inventive technological solution that provides for improved methods and systems for storing and managing data and metadata for a block-level reference-based storage system.

C. U.S. Patent No. 8,131,726

37. On March 6, 2012, the U.S. Patent and Trademark Office duly and lawfully issued U.S. Patent No. 8,131,726 (the "'726 patent"), titled "Generic architecture for indexing document groups in an inverted text index." A true and correct copy of the '726 patent is attached as **Exhibit 3**.

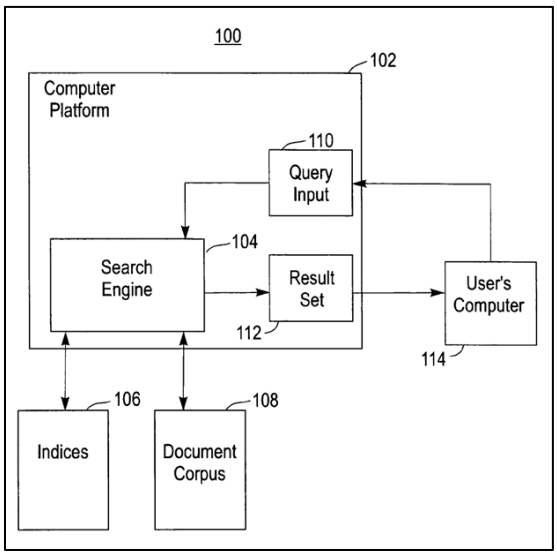
38. Daedalus is the owner and assignee of all right, title, and interest in and to the '726 patent, including the right to assert all causes of action arising under said patent and the right to any remedies for infringement of it.

39. The '726 patent relates to methods and systems of indexing documents for search, where duplicate documents are grouped and indexed only once while still preserving the metadata of each document. (Ex. 3 at Abstract). The purpose of this invention is to speed search by limiting the size of the index while still allowing searches for documents restricted to certain metadata values (e.g. duplicate documents with different creation dates). (Ex. 3 at 1:30-45).

40. The claims of the '726 patent describe elements, individually or as an ordered combination, that were non-routine and unconventional at the time of the invention in 2012 and an improvement over prior art, as they provided a way (not previously available) to index documents for search while maintaining useful metadata for individual documents. For example, it was unconventional to index metadata separately for individual documents while combining the index for duplicate documents.

41. Specifically, the '726 patent seeks to solve a problem in the prior art of indexing increased amounts of data by limiting duplicates while still maintaining robust metadata. (*See* Ex. 3 at 1:13-29). In particular, the specification describes a problem wherein prior art systems would index either all documents separately even if the contents of the files were similar, or in the alternative would index duplicate documents only a single time. (Ex. 3 at 1:30-67). This results in a tradeoff between either compression or data loss from collapsing multiple documents into a single index. The specification describes a number of patents and proposals to solve this problem, all of which were unsuccessful. (Ex. 3 at 2:1-11).

42. An example system that this invention improves is shown in Fig. 1:



(Ex. 3 at Fig. 1).

43. The claims of the '726 solve this tradeoff in a unique way to meet this technological problem, and the patent claims describe novel methodologies that improve the functioning of an indexing and search system. For example, claim 1 recites:

> A method for indexing a plurality of documents, the method comprising the steps of:
>
> a) identifying a duplicate group of documents from among the plurality of documents, each of the documents in the duplicate group comprising respective content and metadata, wherein the respective content of each document in the duplicate group is substantially similar and corresponds to a content for the duplicate group;
>
> b) creating one index of content for the duplicate group;
>
> c) indexing the metadata for each of the documents in the duplicate group;
>
> d) receiving a query and executing said query as if duplicated content was indexed for each document of the duplicate group, and
>
> e) outputting results of said query.

(Ex. 3 at claim 1).

44. In essence, duplicate groups of documents are identified from the documents being indexed. (Ex. 3 at 1:20-24). Content for the duplicate group is indexed only once, however, the metadata for each of the documents in the duplicate group is indexed. (Ex. 3 at 1:24-28).

45. This invention allows duplicate content to be indexed only once, while preserving metadata as if the duplicated content was indexed for each document. (Ex. 3 at 2:256-33). This technological solution improves the functioning of a document storage and search system as it retains the ability to locate documents easily while saving storage space and maximizing performance. (Ex. 3 at 2:30-35). This is a technological solution to a technological problem and presents an inventive concept over the prior art. Indeed, the patent office found that the prior art did not "suggest indexing the *metadata* for each of the documents in *the duplicate group* after the duplicate group of documents are *identified*." 10/3/2011 Decision on Appeal, Appeal 2009-

010638, Application 10/905,604 at 5-6 (emphasis in original); *See also* 10/28/2011 Notice of Allowability, Application 10/905/604 at 2 ("these claims contain limitations that overcome the best possible prior art.").

46. Thus the '726 Patent describes an inventive technological solution that provides for improved methods and systems for maintaining indexing metadata for a search and retrieval system.

47. Each of the Asserted Patents are presumed valid under 35 U.S.C. § 282.

48. Daedalus does not make, offer for sale, or sell within the United States any article practicing the Asserted Patents, or import any article practicing the Asserted Patents into the United States. Daedalus has complied with the requirements of 35 U.S.C. § 287 with respect to the Asserted Patents.

COUNT I

(INFRINGEMENT OF U.S. PATENT NO. 7,542,957)

49. Daedalus incorporates by reference the foregoing paragraphs of this Complaint as if fully set forth herein.

50. Dropbox makes, uses, sells, and/or offers to sell in the United States, and/or imports into the United States products and/or services that utilize the Dropbox API to access Dropbox through other applications.

51. On information and belief, Dropbox has directly infringed and continues to directly infringe one or more claims of the '957 Patent, including at least claim 2 of the '957 Patent, in the state of Delaware, in this judicial district, and elsewhere in the United States by, among other things, making, using, selling, offering for sale, and/or importing into the United States products

that embody one or more of the inventions claimed in the '957 Patent, including but not limited to the Dropbox API, and all reasonably similar products, in violation of 35 U.S.C. § 271(a).

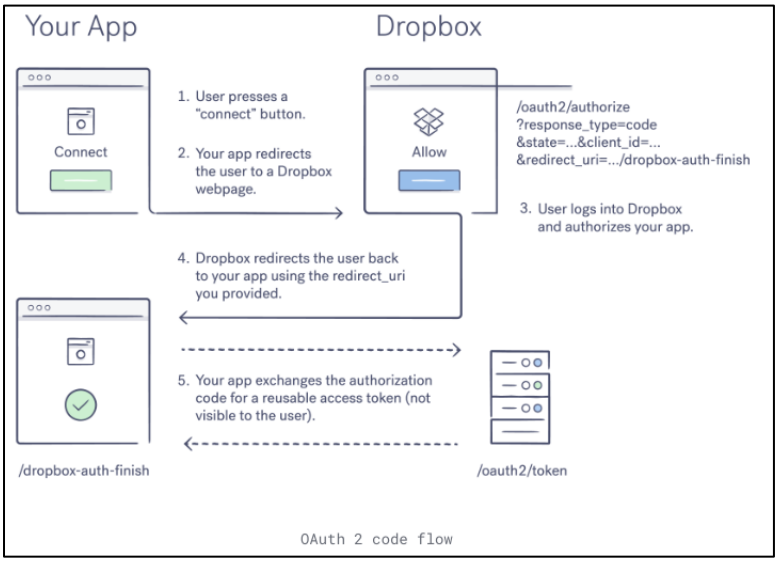
52. As discussed above, the '957 Patent seeks to improve validation for web applications by developing a general-purpose validation process without the need for bespoke code for each individual application. (Ex. 1 at 4:25-35). Since the invention of the '957 Patent, internet security has become paramount. In response, OAuth, which stands for "Open Authorization," was developed as a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user. https://auth0.com/intro-to-iam/what-is-oauth-2. OAuth 2.0 was developed and released in October 2012. https://datatracker.ietf.org/doc/html/rfc6749. OAuth is now the de facto industry standard for online authorization. https://auth0.com/intro-to-iam/what-is-oauth-2. Dropbox has supported OAuth 2.0 since at least September 2013. https://web.archive.org/web/20130921060907/https://www.dropbox.com/developers/core/docs#oa2-authorize. Dropbox's API using OAuth infringes at least claim 2 of the '957 Patent.

53. Dropbox performs a method for validating a request to a Web application, the request having data. Dropbox provides an API that allows third party developers the ability to access and interact with Dropbox within their applications. https://www.dropbox.com/developers/documentation/http/documentation. Dropbox uses Open Authorization Release 2.0 (OAuth 2.0) to validate requests for access:

> Authorization
>
> Dropbox uses OAuth, an industry-standard protocol for authorization, to allow users to grant apps account access without exposing their account credentials. We support OAuth 2.0 for authenticating API requests; requests are authenticated through the Dropbox website or mobile app. Dropbox support OAuth best practices, including short-lived access tokens and PKCE for distributed apps.

https://assets.dropbox.com/www/en-us/business/solutions/solutions/dfb_security_whitepaper.pdf at p. 51.

54. Dropbox's OAuth engine allows other apps to access a user's Dropbox account as long as the request is properly validated:



https://developers.dropbox.com/oauth-guide

55. An authenticated API request from a user's app may interact with files and metadata in the user's Dropbox account, which is accessible via Web application, and perform other functions, including receiving Dropbox account updates via HTTP and invoking other web apps. A request includes data including JSON arguments in an HTTP POST request. https://www.dropbox.com/developers/documentation/http/documentation.

56. An authorization request to Dropbox results in an access token that is provided that determines whether requests are valid for a particular scope:

> User Authentication
>
> This is the most common authentication type. This type uses an access token for a specific user and app pair, in order to operate on that user's account, to the extent allowed by that app's permission. Applications that authorize only scopes for the User API will receive a user access token.
>
> **Example:**

curl -X POST "https://api.dropboxapi.com/2/users/get_current_account" \

--header "Authorization: Bearer <OAUTH2_ACCESS_TOKEN>

https://www.dropbox.com/developers/reference/auth-types.

57. Dropbox creates a validation engine (an OAuth endpoint) to handle OAuth requests. The validation engine comprises logic including validation rules, called OAuth scopes, that relate to a defined plurality of data elements, called OAuth objects.

58. Dropbox follows the OAuth specification which defines what scopes are and how they are used:

> OAuth Scopes
> tools.ietf.org/html/rfc6749#section-3.3
>
> Scope is a mechanism in OAuth 2.0 to limit an application's access to a user's account. An application can request one or more scopes, this information is then presented to the user in the consent screen, and the access token issued to the application will be limited to the scopes granted.
>
> The OAuth spec allows the authorization server or user to modify the scopes granted to the application compared to what is requested, although there are not many examples of services doing this in practice.
>
> OAuth does not define any particular values for scopes, since it is highly dependent on the service's internal architecture and needs.

https://oauth.net/2/scope/

59. Scopes (the validation rules) include actions that are available to a user based on account, including read and write. And the scopes apply to data elements, for example files and folders. When an API call to Dropbox is made, Dropbox's OAuth validation engine validates the scope based on the user's information. Then based on the selected scope, the authorizations for data elements are determined:

> Scopes are generally organized into major actions, often read and write, on major objects. These objects include:

- Account Info

- Files and Folders
- Collaboration (sharing)

For applications using the Business API to operate on Teams, this also includes:

- Team Info
- Membership
- Group Settings
- Sessions
- Team Data
- The Event log

https://developers.dropbox.com/oauth-guide

60. The Dropbox API uses OAuth scopes to determine the actions an application is allowed to perform on a user's data. https://developers.dropbox.com/oauth-guide#diagram

61. Dropbox loads a validation rule (an OAuth scope) when handling requests. Scopes are retrieved on a user or team basis depending on which Dropbox API is being called. Apps using that API may request different scopes.

> Using User and Team Tokens
>
> Apps that request only scopes from the User API will receive a token associated with the user that authorizes the app.
>
> When an app requests team scopes for the Business API, the resulting token is associated with the team (rather than the administrator who authorized it). The user scopes requested as part of the team authorization define the calls that can be used when acting on behalf of a team member with Dropbox-API-Select-User.
>
> Apps may request different scopes, per authorization, using the scopes parameter in the Authorization URL. For example, user apps may re-authorize to enable additional team functionality.

https://developers.dropbox.com/oauth-guide#scopes

62. Dropbox applies the validation rules (OAuth scopes) to the data elements (user files) and sends the request to the web application. Scopes (the validation rules) include actions that are available to a user based on account, including read and write. And the scopes apply to

data objects, for example files and folders. When an API call to Dropbox is made, Dropbox's OAuth validation engine validates the scope based on the user's information. Then based on the selected scope, the authorizations for data elements are determined: "The Dropbox API uses OAuth scopes to determine the actions your application is allowed to perform on a user's data. When you create your application in the App Console, you'll choose from different scopes in the 'Permissions' tab." https://developers.dropbox.com/oauth-guide. "The selected scopes are applied to your access token and determine which API calls your application is allowed to execute. This level of access is then communicated to the end user on the Dropbox app authorization page where the user consents to sharing their data." https://developers.dropbox.com/oauth-guide.

63. As a result, Dropbox performs all elements of at least claim 2 of the '957 Patent.

64. By making, using, offering for sale, and/or selling products in the United States and/or importing products into the United States, Dropbox has injured Daedalus and is liable to Daedalus for directly infringing one or more claims of the '957 Patent, including without limitation claim 2 pursuant to 35 U.S.C. § 271(a).

65. On information and belief, Dropbox will continue to infringe the '957 Patent unless enjoined by this Court.

66. As a result of Dropbox's infringement of the '957 Patent, Daedalus has suffered monetary damages, and seeks recovery, in an amount to be proven at trial, adequate to compensate for Dropbox's infringement, but in no event less than a reasonable royalty with interest and costs.

67. Dropbox's infringement of Daedalus' rights under the '957 Patent will continue to damage Daedalus, causing irreparable harm for which there is no adequate remedy at law, unless enjoined by this Court.

COUNT II

(INFRINGEMENT OF U.S. PATENT NO. 8,176,269)

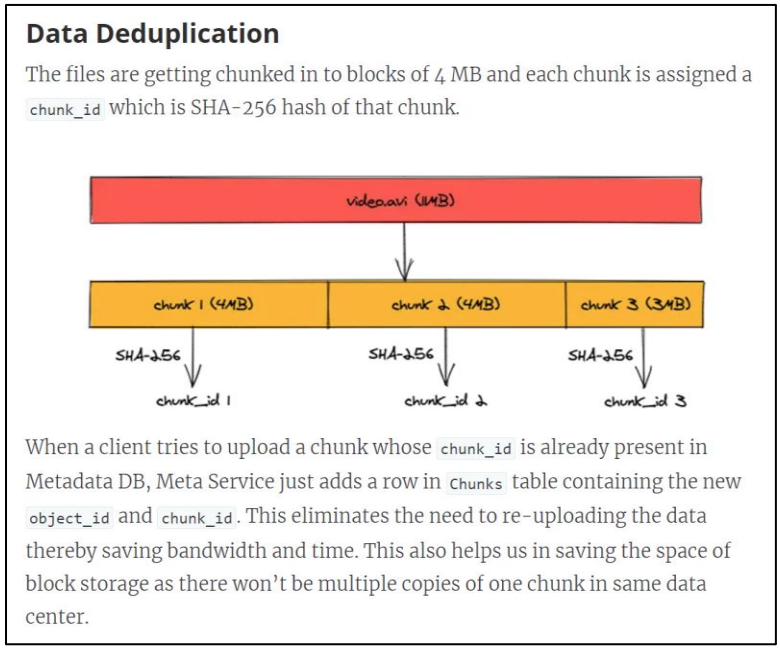
68. Daedalus incorporates by reference the foregoing paragraphs of this Complaint as if fully set forth herein.

69. Dropbox makes, uses, sells, and/or offers to sell in the United States, and/or imports into the United States online data storage products and/or services including ones that use Dropbox's Magic Pocket technology.

70. On information and belief, Dropbox has directly infringed and continues to directly infringe one or more claims of the '269 Patent, including at least claim 1 of the '269 Patent, in the state of Delaware, in this judicial district, and elsewhere in the United States by, among other things, making, using, selling, offering for sale, and/or importing into the United States products that embody one or more of the inventions claimed in the '269 Patent, including but not limited to data storage products using Dropbox's Magic Pocket, and all reasonably similar products, in violation of 35 U.S.C. § 271(a).

71. As discussed above, IBM engineers pioneered a method, for example claim 1 of the '269, for deduplicating data and managing unreferenced data blocks in 2012. (*See* Ex. 2 at claim 1). Later, Dropbox developed a storage system called "Magic Pocket" to improve the performance of Dropbox's file sharing systems and improve its unit economics. https://dropbox.tech/infrastructure/magic-pocket-infrastructure. Since the launch of Magic Pocket in 2014, Dropbox invested significant resources therein to maintain its position as a premier online storage solution and serve as a key product differentiator. *Id.* At its core, Magic Pocket is built on systems and methods that IBM pioneered, and Dropbox's Magic Pocket infringes at least claim 1 of the '269 Patent.

72. Dropbox's Magic Pocket storage system maintains file metadata for files having data blocks in a computer readable storage device, wherein at least one of the files has file metadata indicating that the file has multiple data blocks. Specifically, as described in materials posted online, every file in Dropbox is broken into 4-megabyte chunks (*i.e.*, data blocks) that have an object ID (*i.e.*, data block reference) and chunk ID (*i.e.*, content identifier). *See* https://dropbox.tech/infrastructure/streaming-file-synchronization ("Every file in Dropbox is partitioned into 4MB blocks, with the final block potentially being smaller. These blocks are hashed with SHA-256 and stored. A file's contents can be uniquely identified by this list of SHA-256 hashes, which we refer to as a 'blocklist'.") This data chunking saves space in block storage and saves bandwidth and time:

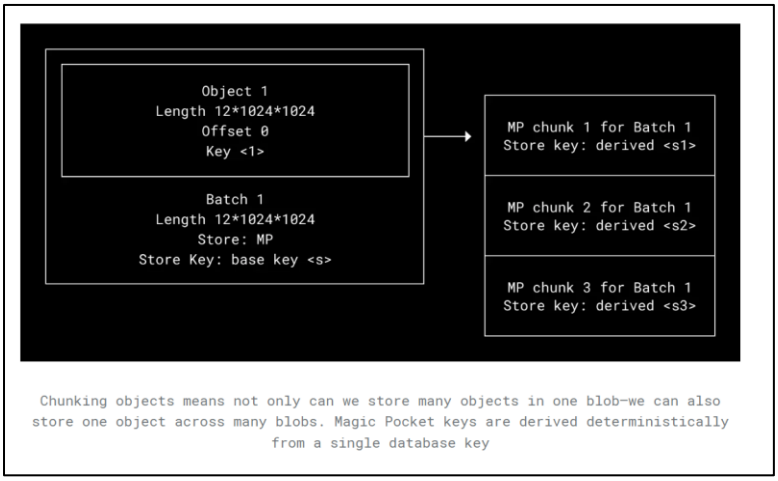


https://systemdesignprimer.com/dropbox-system-design/

73. Dropbox's Magic Pocket maintains data block metadata for each data block, wherein the data block metadata for one data block includes a data block reference and content identifier identifying content of the data block, wherein the file metadata for each file includes the

data block reference to each data block in the file. https://dropbox.tech/infrastructure/inside-the-magic-pocket#put ("When a Put request arrives, the Frontend first checks if the block already exists (via the Block Index) and then chooses a target volume to store the block.")

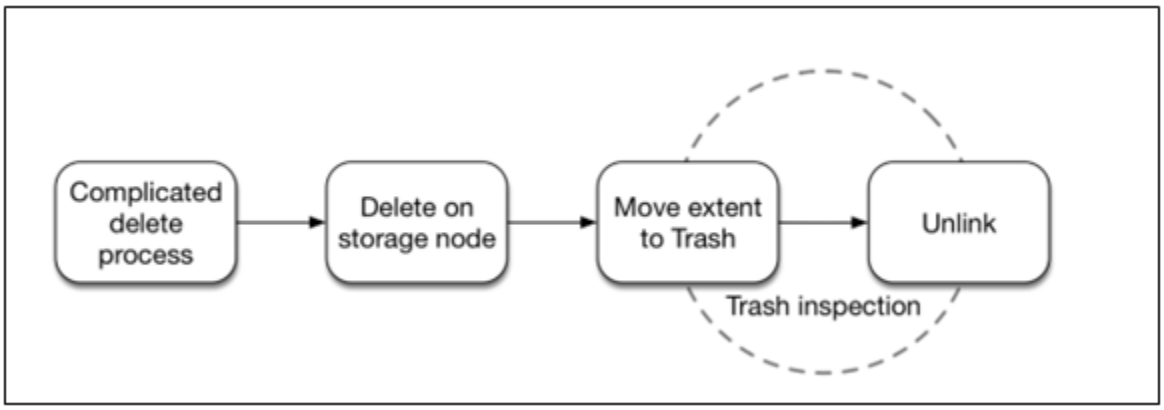
74. Specifically, files metadata resides in the Server File Journal (SFJ), which for each file includes a blocklist referencing the blocks that comprise a file. *Id.* ("[The SFJ] is our big metadata database which represents our file system! Note that it doesn't contain file contents, just blocklists. It is an append-only record where each row represents a particular version of a file.") Files in Magic Pocket are made up of stored keys that correspond to data chunks and a record in the SFJ that references multiple chunks indicating that the file has multiple data blocks associated with it:



https://dropbox.tech/infrastructure/abstracting-cloud-storage-backends-with-object-store#chunking

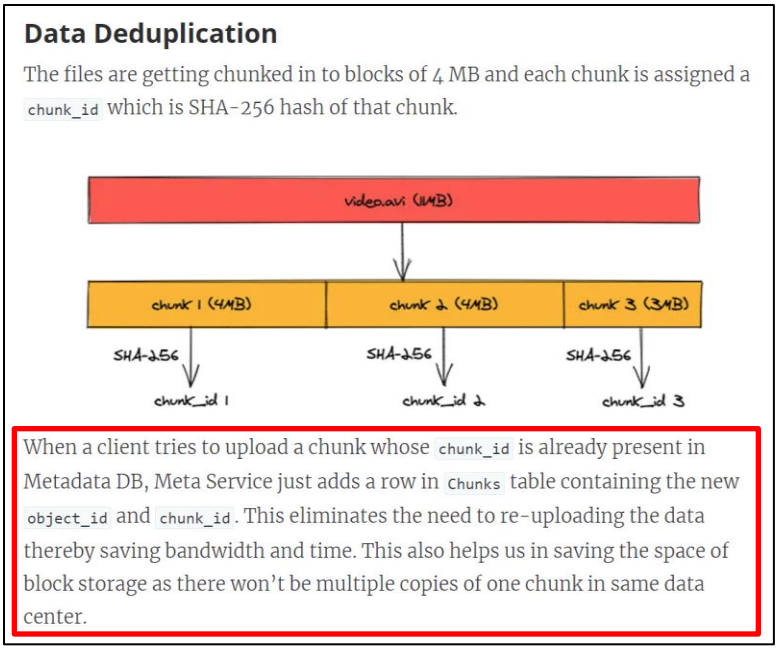
75. Dropbox's Magic Pocket determines that an unreferenced data block has become unreferenced. Specifically, Dropbox determines whether a data block has been moved or deleted via a process involving the "Trash Inspector." https://dropbox.tech/infrastructure/pocket-watch ("If Magic Pocket needs to move a volume between storage nodes, or rewrite a volume after garbage collecting it, then it writes the volume to a new set of storage nodes before deleting it from the old nodes.") "The Trash Inspector iterates over all the blocks in trash extents and checks the

Block Index to determine that either: the block has been safely moved to a new set of storage nodes, or the block itself was also marked to be deleted." https://dropbox.tech/infrastructure/pocket-watch#trash-inspector. The Trash Inspector uses metadata to indicate that the data block is unreferenced:



https://dropbox.tech/infrastructure/pocket-watch#trash-inspector

76. Dropbox adds the data block reference of the unreferenced data block metadata in the computer readable storage device to file metadata for an added file that includes multiple data blocks including one data block having content matching the content of the unreferenced data block according to the content identifier in the unreferenced data block metadata. Specifically, trash and unlinked files remain in Dropbox's Block Index. *See* https://dropbox.tech/infrastructure/pocket-watch#trash-inspector ("This 'trash' data sits there until we can be sure this data was deleted correctly.") These data blocks are available when new files are put on the system or to reconstitute files that were inadvertently deleted:



https://systemdesignprimer.com/dropbox-system-design/ (emphasis added)

77. As a result, Dropbox's Magic Pocket performs all elements of at least claim 1 of the '269 Patent.

78. By making, using, offering for sale, and/or selling products in the United States and/or importing products into the United States, Dropbox has injured Daedalus and is liable to Daedalus for directly infringing one or more claims of the '269 Patent, including without limitation claim 1 pursuant to 35 U.S.C. § 271(a).

79. On information and belief, Dropbox will continue to infringe the '269 Patent unless enjoined by this Court.

80. As a result of Dropbox's infringement of the '269 Patent, Daedalus has suffered monetary damages, and seeks recovery, in an amount to be proven at trial, adequate to compensate for Dropbox's infringement, but in no event less than a reasonable royalty with interest and costs.

81. Dropbox's infringement of Daedalus' rights under the '269 Patent will continue to damage Daedalus, causing irreparable harm for which there is no adequate remedy at law, unless enjoined by this Court.

COUNT III

(INFRINGEMENT OF U.S. PATENT NO. 8,131,726)

82. Daedalus incorporates by reference the foregoing paragraphs of this Complaint as if fully set forth herein.

83. Dropbox makes, uses, sells, and/or offers to sell in the United States, and/or imports into the United States online data storage products and/or services including ones that use Dropbox's Magic Pocket and Nautilus search technology.

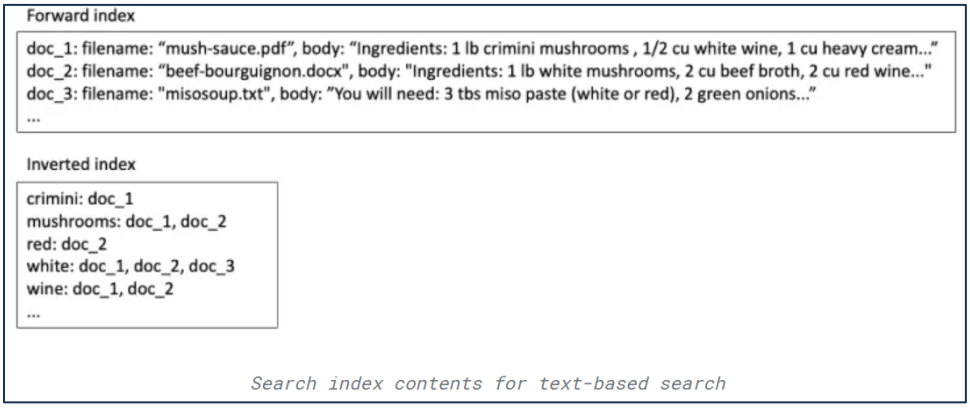
84. On information and belief, Dropbox has directly infringed and continues to directly infringe one or more claims of the '726 patent, including at least claim 1 of the '726 patent, in the state of Delaware, in this judicial district, and elsewhere in the United States by, among other things, making, using, selling, offering for sale, and/or importing into the United States products that embody one or more of the inventions claimed in the '726 patent, including but not limited to data storage products using Dropbox's Magic Pocket and Nautilus search, and all reasonably similar products, in violation of 35 U.S.C. § 271(a).

85. Dropbox's systems hold copious amounts of data and rely on the patented technology of the '726 patent to provide an effective and efficient search function for finding documents. Indeed, Dropbox's document storage and search functionality performs each element of at least claim 1 of the '726 patent.

86. Specifically, Dropbox performs a method for indexing a plurality of documents for search. https://dropbox.tech/machine-learning/architecture-of-nautilus-the-new-dropbox-search-

engine ("The role of the indexing pipeline is to process file and user activity, extract content and metadata out of it, and create a search index. The serving system then uses this search index to return a set of results in response to user queries.") Dropbox's search functionality creates a forward index of file content and metadata, as well as an inverted index that maps each word to a list of files that contain that word. https://dropbox.tech/machine-learning/how-image-search-works-at-dropbox. Specifically, Dropbox's website describes its "Nautilus" functionality:

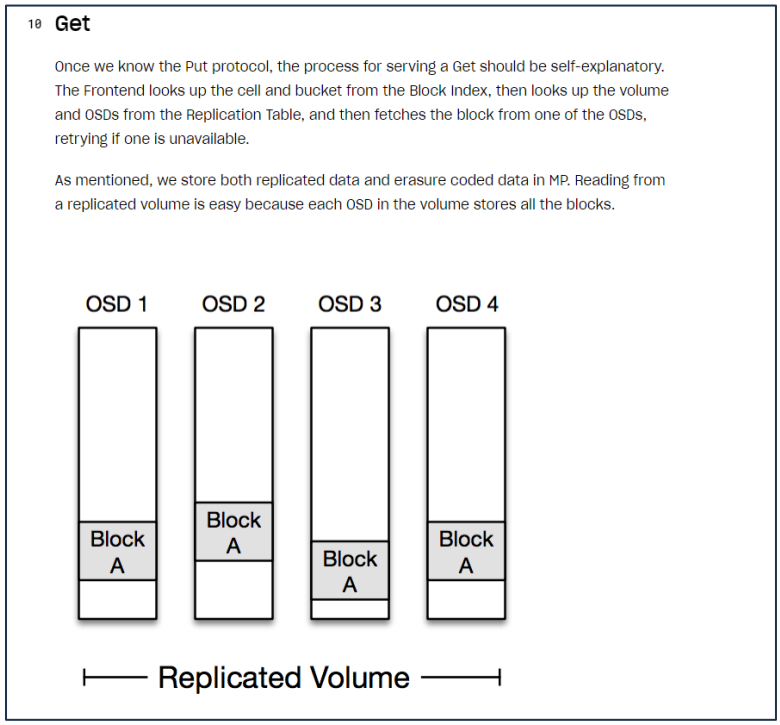
> Conceptually, Nautilus consists of a forward index that maps each file to some metadata (e.g. the filename) and the full text of the file, and an inverted index that maps each word to a posting list of all the files that contain the word. For text-based search, the index content for a few recipe files might look something like this:



https://dropbox.tech/machine-learning/how-image-search-works-at-dropbox

87. The index includes both content of the documents as well as metadata. https://dropbox.tech/machine-learning/architecture-of-nautilus-the-new-dropbox-search-engine#indexing ("What are the kinds of things users would like to search by? Of course there is the content of each document, i.e., the text in the file. But there are also numerous other types of data and metadata that are relevant.") https://dropbox.tech/machine-learning/architecture-of-nautilus-the-new-dropbox-search-engine.

88. Dropbox identifies a duplicate group of documents from among a plurality of documents, each of the documents in the duplicate group comprising respective content and metadata, wherein the respective content of each document in the duplicate group is substantially similar and corresponds to a content for the duplicate group. Specifically, Dropbox generates redundant copies of files that are spread over different storage locations in case of failure. The contents of these files will be substantially similar:

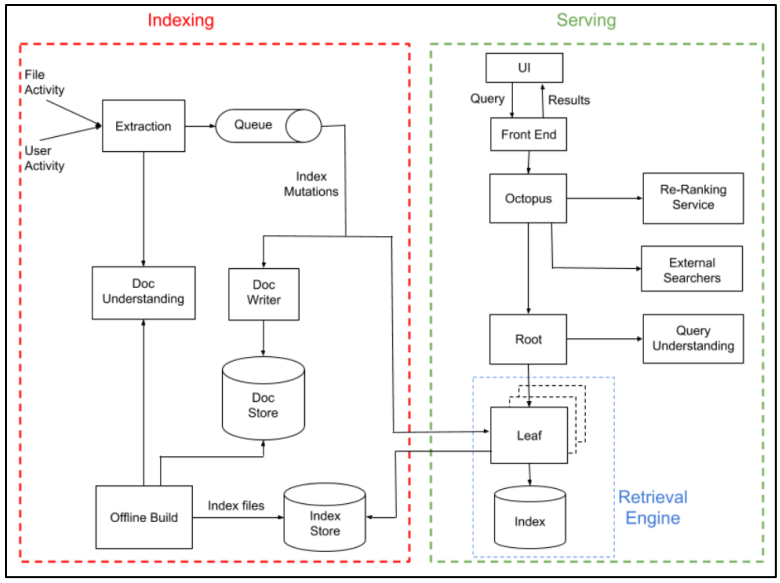


https://dropbox.tech/infrastructure/inside-the-magic-pocket#get

89. As Dropbox's website describes, "[e]ach block in [Magic Pocket] is stored independently in at least two separate zones and then replicated reliably within these zones. This redundancy is great for avoiding natural disasters and large-scale outages but also allows us to establish very clear administrative domains and abstraction boundaries to avoid a misconfiguration or congestion collapse from cascading across zones." The documents stored by Magic Pocket include both file content and metadata about the files.

90. The documents stored by Magic Pocket include both file content and metadata about the files. Dropbox creates one index of content for the duplicate group. Specifically, Dropbox indexes the file once even though it is replicated across locations. https://dropbox.tech/infrastructure/inside-the-magic-pocket ("Dropbox stores two kinds of data: file content and metadata about files and users. Magic Pocket is the system we use to store the file content. These files are split up into blocks, replicated for durability, and distributed across our infrastructure in multiple geographic regions.") Dropbox also indexes the metadata for each of the documents in the duplicate group. *Id.*; *see also* https://www.dropbox.com/business/trust/security/architecture ("File metadata is stored in a MySQL-backed database service, and is sharded and replicated as needed to meet performance and high availability requirements.") In this way, Dropbox maintains references to each of the replicated documents in order to find them. Moreover, when a user makes changes to a file on Dropbox, the content of the data blocks is not changed, but a separate system called FileJournal maintains metadata for that block. https://dropbox.tech/infrastructure/inside-the-magic-pocket. Likewise, a separate set of metadata is kept on a replicated block level in the ReplicationTable. *Id.*

91. Finally, Dropbox uses a retrieval engine that accepts queries for documents and outputs the results of the file that is searched for a single time, despite it being replicated:



https://dropbox.tech/machine-learning/architecture-of-nautilus-the-new-dropbox-search-engine

92. As Dropbox itself describes Nautilus, "[t]he retrieval engine is a distributed system which fetches documents that match a search query. The engine is optimized for performance and high recall—it aims to return the largest set of candidates possible in the given allocated time budget." https://dropbox.tech/machine-learning/architecture-of-nautilus-the-new-dropbox-search-engine. As a result, Dropbox keeps a single index for a file while also maintaining metadata relating to the duplicates. Dropbox therefore utilizes the patented technology of the '726 patent to provide enhanced search for documents on its systems.

93. By making, using, offering for sale, and/or selling products in the United States and/or importing products into the United States, Dropbox has injured Daedalus and is liable to Daedalus for directly infringing one or more claims of the '726 patent, including without limitation claim 2 pursuant to 35 U.S.C. § 271(a).

94. On information and belief, Dropbox will continue to infringe the '726 patent unless enjoined by this Court.

95. As a result of Dropbox's infringement of the '726 patent, Daedalus has suffered

monetary damages, and seeks recovery, in an amount to be proven at trial, adequate to compensate for Dropbox's infringement, but in no event less than a reasonable royalty with interest and costs.

96. Dropbox's infringement of Daedalus' rights under the '726 patent will continue to damage Daedalus, causing irreparable harm for which there is no adequate remedy at law, unless enjoined by this Court.

PRAYER FOR RELIEF

WHEREFORE, Daedalus prays for judgment and seeks relief against Dropbox as follows:

A. For judgment that Dropbox has infringed and continues to infringe the claims of the Asserted Patents;

B. For judgment awarding Daedalus damages adequate to compensate it for Dropbox's infringement of the Asserted Patents, including all pre-judgment and post-judgment interest as well as an award of mandatory future royalties for continuing infringement;

C. For a permanent injunction against Dropbox and its respective officers, directors, agents, servants, affiliates, employees, divisions, branches, subsidiaries, parents, and all other acting in active concert therewith from infringement of the Asserted Patents;

D. For judgment awarding enhanced damages pursuant to 35 U.S.C. § 284;

E. For judgment awarding attorneys' fees pursuant to 35 U.S.C. § 285 or otherwise permitted by law;

F. For judgment awarding costs of suit; and

G. For judgment awarding Daedalus such other and further relief as the Court may deem just and proper.

DEMAND FOR JURY TRIAL

Pursuant to Rule 38(b) of the Federal Rules of Civil Procedure Daedalus hereby demands a trial by jury of this action.

Dated: August 30, 2024

By: */s/ Stephen B. Brauerman*
Stephen B. Brauerman
Ronald P. Golden III
BAYARD, P.A.
600 N. King Street, Suite 400
P.O. Box 25130
Wilmington, DE 19899
302-655-5000
sbrauerman@bayardlaw.com
rgolden@bayardlaw.com

Of Counsel:

Denise M. De Mory
Richard Lin
Michael E. Flynn-O'Brien
Gareth DeWalt
BUNSOW DE MORY LLP
701 El Camino Real
Redwood City, CA 94063
Tel.: (650) 351-7248
Fax: (415) 426-4744
ddemory@bdiplaw.com
rlin@bdiplaw.com
mflynnobrien@bdiplaw.com
gdewalt@bdiplaw.com

Counsel for Plaintiff