IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Inter Partes Review of: )
U.S. Patent No. 8,566,361 )
Issued: October 22, 2013 )
Application No.: 13/316,263 )
Filing Date: December 9, 2011 )

For: Datacenter Workflow Automation Scenarios Using Virtual Databases

FILED VIA PRPS

PETITION FOR INTER PARTES REVIEW
OF U.S. PATENT NO. 8,566,361

For ease of reference, Petitioner refers to this Petition as “‘361 Petition I,”
challenging claims 1-6, 8, 14, 16-19, 24 and 25.
# Table of Contents

I. INTRODUCTION ........................................................................................................... 1

II. REQUIREMENTS FOR PETITION FOR *INTER PARTES REVIEW* .......... 1

   A. Grounds for Standing (37 C.F.R. § 42.104(a)) ........................................ 1
   B. Notice of Lead and Backup Counsel and Service Information ......... 1
   C. Notice of Real-Parties-in-Interest (37 C.F.R. § 42.8(b)(1)) .......... 2
   D. Notice of Related Matters (37 C.F.R. § 42.8(b)(2)) ....................... 2
   E. Fee for *Inter Partes* Review .............................................................. 3
   F. Proof of Service ............................................................................................. 3

III. IDENTIFICATION OF CLAIMS BEING CHALLENGED

     (§ 42.104(B)) ............................................................................................................. 3

IV. DESCRIPTION OF THE PURPORTED INVENTION ............................................ 3

V. CLAIM CONSTRUCTION ............................................................................................. 6

   A. Applicable Law ................................................................................................. 6
   B. Construction of Claim Terms ............................................................................. 6

VI. PERSON HAVING ORDINARY SKILL IN THE ART ............................................. 10

VII. THE NETAPP PRIOR ART ...................................................................................... 10

   A. Overview Of The NetApp Prior Art ............................................................ 11
   B. Sanders (Ex. 1004) ......................................................................................... 14
   C. Edwards (Ex. 1005) ......................................................................................... 17
   D. Chapman (Ex. 1006) ....................................................................................... 22

VIII. MOTIVATIONS TO COMBINE THE NETAPP PRIOR ART

    REFERENCES ............................................................................................................. 23

    A. Motivation To Combine References From The Same Company
       That Describe The Same Product ................................................................. 24
    B. Motivation to Combine Sanders with Edwards and Chapman .......... 24

IX. PRECISE REASONS FOR THE RELIEF REQUESTED .......................................... 27

    A. Ground 1: Claims 1-6, 8, 14, 16-19, 24, and 25 are rendered
       obvious over Sanders in view of Edwards and Chapman. .................... 28
Exhibit List

1001  U.S. Patent No. 8,566,361 (‘‘361 patent’’)
1002  File History for the ‘361 Patent
1003  File History for U.S. Patent No. 8,161,077 (‘‘077 Patent’’)
1008  Delphix Corporation’s Infringement Contentions for the ‘361 Patent
1009  FlexClone, NetApp, Inc. Datasheet
1010  FlexVol™ and FlexClone™ Software, NetApp, Inc. Datasheet
1011  NetApp SnapManager for Oracle, NetApp, Inc. Datasheet
1012  NetApp SnapMirror, NetApp, Inc. Datasheet
1013  Network Appliance™ Snapshot™ Technology, NetApp, Inc., Datasheet
1014  LinkedIn Employment Profiles for NetApp, Inc. Employees
1015  Erez Zadok, Curriculum Vitae
1016  Declaration of Erez Zadok In Support of Petition I For Inter Partes Review of U.S. Patent No. 8,566,361 (“Zadok Decl.”)
1018  Delphix Corporation’s Claim Construction Proposals
I. INTRODUCTION

On behalf of Actifio, Inc. ("Petitioner") and in accordance with 35 U.S.C. § 311 and 37 C.F.R. § 42.100, *inter partes* review of claims 1-6, 8, 14, 16-19, 24 and 25 of United States Patent No. 8,566,361, titled “Datacenter Workflow Automation Scenarios Using Virtual Databases” (the “‘361 patent”) is hereby requested. According to USPTO records, the ‘361 patent is currently assigned to Delphix Corp. (“Delphix”). A copy of the ‘361 patent is provided as Ex. 1001, and its prosecution history as Ex. 1002.

II. REQUIREMENTS FOR PETITION FOR *INTER PARTES REVIEW*

A. Grounds for Standing (37 C.F.R. § 42.104(a))

Petitioner certifies that the ‘361 patent is available for *inter partes* review and that Petitioner is not barred or estopped from requesting *inter partes* review of the challenged claims of the ‘361 patent on the grounds identified herein.

B. Notice of Lead and Backup Counsel and Service Information

Pursuant to 37 C.F.R. §§ 42.8(b)(3), 42.8(b)(4), and 42.10(a), Petitioner provides the following designation of Lead and Back-Up counsel.

<table>
<thead>
<tr>
<th>LEAD COUNSEL</th>
<th>BACKUP COUNSEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robert Steinberg (Reg. No. 33,144) (<a href="mailto:bob.steinberg@lw.com">bob.steinberg@lw.com</a>)</td>
<td>Jonathan Link (Reg. No. 41,548) (<a href="mailto:jonathan.link@lw.com">jonathan.link@lw.com</a>)</td>
</tr>
<tr>
<td>Postal &amp; Hand-Delivery Address: Latham &amp; Watkins LLP 355 South Grand Avenue</td>
<td>Postal &amp; Hand-Delivery Address: Latham &amp; Watkins LLP 555 Eleventh St., N.W., Suite 1000</td>
</tr>
</tbody>
</table>
Pursuant to 37 C.F.R. § 42.10(b), a Power of Attorney is attached hereto.

C. **Notice of Real-Parties-in-Interest (37 C.F.R. § 42.8(b)(1))**

The real-parties-in-interest is Actifio, Inc. No other party exercised or could have exercised control over this petition; no other party funded or directed this petition. (*See* Office Patent Trial Practice Guide, 77 Fed. Reg. 48759-60.)

D. **Notice of Related Matters (37 C.F.R. § 42.8(b)(2))**

records, application 14/058,873 claims the benefit of the priority of the filing date of the ’361 patent.

E. Fee for Inter Partes Review
The Director is authorized to charge the fee specified by 37 CFR § 42.15(a) to Deposit Account No. 506269.

F. Proof of Service
Proof of service of this petition on the patent owner at the correspondence address of record for the ‘361 patent is attached hereto.

III. IDENTIFICATION OF CLAIMS BEING CHALLENGED
($42.104(B))
Claims 1-6, 8, 14, 16-19, 24, and 25 of the ‘361 patent (the “challenged claims”) are unpatentable in view of the following prior art.

- Sanders et al., DB2: Cloning a Database using NetApp FlexClone™ Technology, April 30, 2006 (“Sanders,” Ex. 1004);

Specifically, claims 1-6, 8, 14, 16-19, 24, and 25 are invalid under 35 U.S.C. § 103 as obvious over Sanders in view of Edwards and Chapman.

IV. DESCRIPTION OF THE PURPORTED INVENTION
In many organizations, day-to-day transactions are run on production databases. ‘361 patent at 1:31-33. To protect the production environment, copies of production databases are used for testing changes to the production databases. *Id.* at 1:33-36. The lifecycle of a change involves various stages, such as development, testing, certification, and training stages. *Id.* at 1:39-43. Creating a database copy for each stage requires redundant and expensive hardware infrastructure and time overhead to copy the data, leading to high space, power, and cost requirements and inefficient use of available resources. *Id.* at 1:47-51.

The purported invention of the ‘361 patent solves these problems by creating “virtual databases,” which are instantaneous database copies that occupy little additional space. See Zadok Decl. at ¶ 21. Virtual databases and associated operations, such as linking, loading, and provisioning, may be used in the workflow scenarios typically executed with conventional database systems. *Id.*

A system “mirrors” the production database by creating copies of the production database at various points-in-time and sending those copies to a remote storage system. *Id.* at ¶¶ 22-23. The initial “baseline” transfer is a point-in-time copy of all of the data in the database; subsequent point-in-time copies (snapshots) transfer and store only “changed” database information. *Id.* The system creates any number of read/writable copies of the production database based on the snapshots, which represent the virtual databases. *Id.* at ¶ 24. These point-in-time copies may
then be mounted on separate systems, such as “virtual database” systems. *Id.*

FIG. 10 of the ‘361 patent illustrates the file structure 1050 created on the remote storage system and representing a set of files for a virtual database:

![Diagram](image)

The file structure’s blocks V11, V12 . . . V2N are “implemented as **pointers**”1 to the actual database block that stores the data.” ‘361 patent at 19:46-49; Zadok Decl. at ¶ 25. The copies of the production database for mounting are “virtual” because data that is common across different copies is not unnecessarily duplicated for each copy. *Id.* at ¶ 24. Instead, the virtual databases each point to the same underlying data stored in the storage system, where data is common over time. *Id.*

The purported invention’s efficiency gains result from this space-efficient creation of read/writable virtual database copies of the production database. *Id.*

---

1 All emphases added unless otherwise noted.
Because of these properties, virtual databases can be created quickly and with minimal storage space requirements beyond the space required for the underlying database snapshots. *Id.* at ¶ 36. The difference is that the files in the virtual database can be written to, whereas the point-in-time copy is a read-only version. *Id.* at ¶ 38. The ‘361 patent describes leveraging the virtual database systems for various workflow scenarios including database replication, backup and recovery, and data warehouses and data marts. *Id.* at ¶ 42; see also ¶¶ 39-41.

V. CLAIM CONSTRUCTION

A. Applicable Law

In deciding whether to institute *inter partes* review, “[a] claim in an unexpired patent shall be given its broadest reasonable construction in light of the specification of the patent in which it appears.” 37 C.F.R. § 42.100(b). To the extent there is any ambiguity regarding the “broadest reasonable construction” of a claim term, the ambiguity should be resolved in favor of the broader construction absent amendment by the patent owner. Final Rules, 77 Fed. Reg. at 48699.

B. Construction of Claim Terms

Petitioner expressly reserves the right to challenge one or more claims (and claim terms) of the ‘361 patent for failure to satisfy the requirements of 35 U.S.C. § 112, which cannot be raised in these proceedings. *See* 35 U.S.C. § 311(b).
All claim terms not specifically addressed in this section have been accorded their broadest reasonable interpretation as understood by one of ordinary skill in the art and consistent with the specification of the ‘361 patent. Petitioner submits that the following terms may need to be construed in connection with this IPR:

1. **“virtual database”**

   The term “virtual database” is used in challenged claims 1, 4, 6, 8, 14, 17 and 24. The specification explains “virtual databases are ‘virtual’ in the sense that the physical implementation of the database files is decoupled from the logical use of the database files by a database server.” ‘361 patent at 6:19-22. The ‘361 patent discloses a method for creating a virtual database that involves creating files by creating a file structure that includes pointers to the database blocks of a point-in-time copy of a source database. Zadok Decl. at ¶ 98.

   For example, Claim 1 of the ‘361 patent specifically calls for receiving and storing database blocks for the point-in-time copies of the source database. ‘361 patent at Claim 1. It further requires “creating a set of files linked to the stored database blocks.” Id. In addition, the patent states that this “link” of the files to the database blocks is achieved by using pointers. Zadok Decl. at ¶ 100. For instance, the file structure 1050 in Figure 10 of the ‘361 patent is created with its database blocks “implemented as pointers to the actual database block that stores the data.” Id. at 19:46-48, 51-53 (“V11 points at F11. …V12 points at the block F22.”).
Moreover, both the claims and the specification make clear that the set of files in a virtual database must be capable of being “read from and written to.” See ‘361 patent, Claim 1 (“... allowing the database server running on the system to access the set of files”); id. at 6:63-65 (“The database server on which a virtual database has been provisioned can then read from and write to the files stored on the storage system.”).

Accordingly, the broadest reasonable construction of the term “virtual database” means “a set of database files capable of being read from and written to, created by pointing to already-stored database blocks.”\(^3\) Zadok Decl. at ¶ 102.

2. **“database blocks”**

The term “database blocks” is used in challenged claims 1-3, 5-6, 14, 16-19, 24, and 25, and refers to the “database blocks” used in the specification. Zadok Decl. at ¶ 104. The specification states that each file of a virtual database “includes

\(^3\) Delphix construes “virtual database” as “A set of files to which a database server can read and write such that the physical implementation of the database files is decoupled from the logical use of the database files by the database server.” Ex. 1018 at 3. This is unhelpful and does not inform the Board how virtual databases are created according to the ‘361 patent. Regardless, Edwards meets Delphix’s construction. See § VII.C.
a set of database blocks and the data structures for referring to the database
blocks.” ‘361 patent at 6:31-35. Accordingly, the broadest reasonable construction
of the term “database blocks” means “a unit of data used by a database.” Zadok
Decl. at ¶ 107.

3. “mounting”

The term “mount[ing]” is used in challenged claims 1, 8, 14, 17, and 24. The
specification teaches that “mounting” the set of files of a virtual database allows
“the database server to read from and write to the set of files.” ‘361 patent at
Abstract. Moreover, “The file sharing manager 370 uses the file sharing system
120 for sharing files. An example of a system for sharing files is a network file
system (NFS).” Id. at 11:35-37. Accordingly, “mounting” a set of files enables
files to be accessible over a file-sharing protocol. Zadok Decl. at ¶ 108. Thus, the
broadest reasonable construction of the term “mounting” means “making
accessible” [to a database server]. Id. at ¶ 109.⁴

⁴ Delphix’s construction of a “database block” is nearly identical: “A unit of data
used by a database and comprises a specific number of bytes stored in the storage.”
Ex. 1018 at 3.

⁵ Delphix’s construction of a “mount[ing]” is nearly identical: “make[ing] available
for access.” Ex. 1018 at 4.
4. “standby database”

The term “standby database” is used in challenged Claim 4. The specification explains that a standby database can be used when another database is down: “the database storage system 2105 can also be used in a high availability scenario where it acts as a standby system that can be used when the source database system 1905 is down.” ‘361 patent at 30:1-4. Accordingly, the broadest reasonable construction of the term “standby database” means “a database (including a virtual database) that may be used in lieu of another database that is unavailable.” Zadok Decl. at ¶ 112.

VI. PERSON HAVING ORDINARY SKILL IN THE ART

For purposes of this review, a “person of ordinary skill in the art” (“POSITA”) is: a person with a Bachelor-level degree in computer science, computer engineering, electrical engineering, or a related field in computing technology, and a few years of experience with data storage systems, or equivalent research experience or knowledge. See Zadok Decl. at ¶ 17. This description is approximate, and a higher level of education or skill might make up for less experience, and vice-versa. Id.

VII. THE NETAPP PRIOR ART

For the Board’s ease of reviewing the prior art references, the declarant, Dr. Zadok, has provided a glossary mapping terms in the ‘361 patent to corresponding terms used by the prior art references discussed herein. Zadok Decl. at § IX.
A. Overview Of The NetApp Prior Art

The purported invention of the ‘361 patent relates to the creation, storage, and mounting of virtual databases for use in workflow scenarios such as backup and recovery and data warehouse building. See ‘361 patent at Abstract; Zadok Decl. at ¶ 43. However, NetApp had already invented and commercialized the technology for creating virtual databases and using virtual databases for such workflows years before the filing date of the earliest patent application leading to the ‘361 patent. Id. The technology was NetApp’s “FlexVol Architecture,” specifically its FlexClone feature, and NetApp used the term “clone database” to refer to what the ‘361 patent calls a “virtual database.” Id. FlexClone, by using pointers to data blocks instead of physically duplicating data blocks, added “a level of indirection, or virtualization between the logical storage space used by a volume and the physical storage space provided by the RAID subsystem.” Edwards at 129.

Generally, a NetApp virtual (clone) database is created by using FlexClone to clone a “snapshot” of the source volume (or FlexVol volume) on which a source database resides. Zadok Decl. at ¶ 44. A snapshot is a point-in-time “read only” copy of a volume. Edwards at 134. For example, in order to clone a database residing on a volume of a production system, one would create a snapshot of the volume and clone that snapshot using FlexClone. Zadok Decl. at ¶ 44; see Sanders at 28. The resulting cloned volume (called a FlexClone volume) would contain a
clone (virtual) database. *Id.* A FlexClone volume “uses space very efficiently, allowing both the original FlexVol volume and the FlexClone volume to share common data, storing only the data that changes between the original volume and the clone.” Sanders at 3. “This provides a huge potential saving in storage space, resources, and cost.” *Id.*

The difference between a snapshot and a FlexClone volume is that a FlexClone volume (and any clone databases contained therein) is writable. Edwards at 134-135. Read-only snapshots were commonly known and used (including by NetApp) to recover files from volumes as they existed at previous points in time. See Zadok Decl. at ¶ 45. By using FlexClone on a snapshot, writable clone (virtual) databases could be created, as was desired. Sanders at 3.

Another NetApp feature called SnapMirror allowed snapshots to be duplicated to a remote destination, effectively “mirroring” a source volume on a source system to a remote, destination system. Zadok Decl. at ¶ 46. Multiple snapshots of a single volume can be mirrored to a destination storage system as the source volume changes over time; the destination storage system receives and stores all of the snapshots, which are read-only copies just like the source snapshots. *Id.* FlexClone could then be used on such mirrored snapshots just like it can be used on any snapshot. *Id.* Thus, by taking a snapshot of a source volume, mirroring it to a destination storage system, and cloning the mirrored snapshot,
NetApp allowed the creation of a writable clone (virtual) database in a storage system based on a production database residing on a production system. Sanders at 3. NetApp also enables the use of a SnapMirror “cascade,” where data is copied not just from a source to a destination, but also from that destination to another destination, and so on in a series. Chapman at 22; Zadok Decl. at ¶ 47.

Sanders describes the entire process of creating and mounting a virtual database on a storage system, based on a source database residing on a remote source system, and is summarized to include the following steps: (1) create a snapshot copy of the FlexVol volume containing the source database (Sanders at 27 (“Create Snapshot copies of the FlexVol volumes.”)); (2) use SnapMirror to mirror the snapshot from the source system to the (destination) storage system (id. (“Update the SnapMirror destination.”)); (3) use FlexClone to create a FlexClone volume based on the mirrored snapshot located at the storage system (id. at 28 (“Create clone volumes using Snapshot copies.”)). The FlexClone volume contains the new clone (virtual) database; and (4) mount the clone (virtual) database on a database server (id. (“Mount the cloned volumes.”)). Zadok Decl. at ¶ 49.

The advantages gained by NetApp’s “database cloning” as described by Sanders are nearly identical to those of a “virtual database” as described by the ‘361 patent. See Zadok Decl. at ¶ 50 (containing summary comparison table).

1. ‘361 Patent Prosecution History
The ‘361 patent is a continuation of U.S. Patent No. 8,161,077 (“‘077 patent”). During prosecution of the ‘077 patent, the applicants submitted five high-level NetApp marketing brochures to the USPTO in an Information Disclosure Statement. Exs. 1009-1013. None of those brochures are longer than two pages. See id. These brochures are marketing materials that omit technical and operational details, and while the brochures disclose individual NetApp features, e.g., SnapShot, SnapMirror, and FlexClone, none explain how these features can be used sequentially and in conjunction with one another to create a virtual database on remote storage systems. Zadok Decl. at ¶ 51. For probably these reasons, the Examiner did not cite these features against the claims of the ‘361 and ‘077 patents when they was pending. Id.

B. Sanders (Ex. 1004)

Sanders is a NetApp, Inc. technical report (TR-3460), published by NetApp, Inc., on April 30, 2006, that describes the benefits of using NetApp technologies, such as FlexClone, FlexVol, and SnapMirror, to create clones of IBM DB2 production databases for disaster recovery, data warehousing, and other common workflows. Zadok Decl. 53. Sanders is prior art to the ‘361 patent under 35 U.S.C. § 102(b), and was not before the USPTO during prosecution of the ‘361 patent.

Sanders teaches the use of database cloning to “provide near-production data for business needs such as application development, QA testing, and report
generation.” Sanders at 3. In traditional database replication, where an entire new copy of the database is created, the database administrator faces challenges such as system downtime, degraded system performance during cloning, maintenance overhead related to frequent data refreshes, and high storage space requirements for the storing of the copy. See id.

Sanders further teaches “key business requirements that can benefit greatly from a clone database created using FlexClone technology.” Sanders at 5. These applications take advantage of FlexClone’s ability to quickly and efficiently create copies of a production database, and include the following:

- **Database Software and Application Upgrade Test**: Cloned databases may be used for testing of application upgrades and software patches before implementing into production. See id.

- **Data Warehouse/Data Mart**: One can create a clone of replicated data at a disaster recovery site or production database to provide a staging area that can be refreshed quickly and used for cleansing and transformation of data. See id.

- **Standby Database**: If a production database is corrupted, work may immediately resume on a cloned database. See id. at 6.

- **Online Database Backup**: The clone database is an “image of the database file system at the time of clone creation.” Id. As such, applications can point directly to the clone database. See id.
Sanders describes the use of FlexClone technology, a feature in NetApp’s Data ONTAP platform, for creating point-in-time copies of a production database nearly instantly. Sanders at 3. “A FlexClone volume is a writable point-in-time image of a FlexVol volume or another FlexClone volume.” Id. A clone database, created using FlexClone technology “is a frozen image of the database file system at the time of the clone creation.” Sanders at 6. A FlexClone volume “uses space very efficiently, allowing both the original FlexVol volume and the FlexClone volume to share common data, storing only the data that changes between the original volume and the clone.” Id. “This provides a huge potential saving in storage space, resources, and cost.” Id; Zadok Decl. at ¶ 56. Further, “a FlexClone volume has all the features and capabilities of a regular FlexVol volume.” Sanders at 3; Zadok Decl. at ¶ 58.

Clones can be created on the same or different storage systems. See Sanders at 6. A clone from a source storage system can be created onto a different destination storage system, using NetApp’s SnapMirror technology. “A SnapMirror source and its corresponding destination can reside on the same storage system or on two separate storage systems that are miles apart . . . .” Id. at 3. Sanders’s Figure 3 (annotated to add source and clone database labels) shows how the SnapMirror technology is used to transmit point-in-time copies of a production database to a destination storage system, and clone databases are
created on the destination storage system.

Id. at 8; Zadok Decl. at ¶ 59. Finally, “[i]n order to access the clone database,” Sanders teaches commands and operations to “mount the clone volumes to a database server.” Sanders at 28-29; Zadok Decl. at ¶ 60.

C. Edwards (Ex. 1005)

Edwards, entitled “FlexVol: Flexible, Efficient File Volume Virtualization in WAFL,” is a paper presented at the 2008 USENIX Annual Technical Conference on June 22-27, 2008, is prior art under § 102(b), and was not before the USPTO during prosecution of the ‘361 patent. Edwards describes in a fair degree of technical detail the NetApp features used, in conjunction with one another, to create the clone file systems of Sanders. Zadok Decl. at ¶ 62. Specifically, Edwards describes three NetApp technologies used by Sanders: snapshots (point-in-time copies of a source file system), SnapMirror (used by a
destination storage system to receive snapshot copies from a source system); and FlexClone (used to create a virtual file system from a read-only snapshot stored on a destination storage system). *Id.*

Edwards describes NetApp’s file system (called WAFL) and its buffer tree structure. Zadok Decl. at ¶ 63. A file system is the code and data structures that implement a persistent hierarchy of files and directories. Edwards at 129-130. A volume is an instantiation of the file system. *Id.* An “aggregate” consists of one or more RAID disks (physical disks) on which volumes may reside. *Id.* at 131. The data structures used by NetApp are shown in Edwards’s Figure 1:

![WAFL data structures](image)

*Figure 1: WAFL data structures*

Each file in a volume comprises data blocks and inodes, which contains pointers to data blocks of the file. *Id.* at 130. The inodes and the data blocks of a file system form a tree, whose root block is called the `vol_info` block. *Id.*; Zadok Decl. at ¶¶ 64-67.
Snapshots—or point-in-time copies—are easy to create in NetApp. By creating new file structures—a copy of the root, vol_info block, which contains pointers to existing data blocks—it is possible to create a snapshot copy of the file system at the same time. Id. at ¶ 69. These snapshots are virtual because they are created by file structures pointing to existing data blocks rather than making physical copies of the data blocks. Id. Edwards also describes a technology called Volume SnapMirror, which mirrors the contents of a volume from a source system to a destination system. Edwards at 133. Mirroring may be performed periodically so that the destination system is updated with changes that occur over time to the source system. SnapMirror keeps the remote (mirrored) volume up to date by advancing its state from snapshot copy to snapshot copy. Id.; Zadok Decl. at ¶ 71.

Edwards further discloses how NetApp uses a container file to achieve its virtualization in what NetApp calls a “FlexVol” volume. The FlexVol container file includes a “container map” that provides block number mappings – i.e., that “implement[s] a level of indirection between physical storage containers (called aggregates) and logical volumes (FlexVol volumes).” Edwards at 130; see also id. at 131 (FlexVol “introduces a level of indirection, or virtualization between the logical storage space used by a volume and the physical storage space.”). The ‘361 patent calls for exactly the same thing, that “the physical implementation of the database files is decoupled from the logical use of the database files.” ‘361 patent
at 6:19-23; Zadok Decl. at ¶ 70.

In database environments, for example, it is often desirable to make writable copies of a production database for development or test purposes.” Edwards at 134. Edwards discloses how NetApp’s FlexClone technology is used to create a writable, virtual copy of a database, which creates a “FlexVol volume in which the active file system is a logical replica of a Snapshot copy in a different FlexVol volume.” Id. at 134. The source volume or “parent volume can be any FlexVol volume, including a read-only mirror.” Id. That is, the source volume can be a read-only copy residing in a SnapMirror destination. The clone, or FlexClone volume, is a full-fledged FlexVol volume with all the features and capabilities of a normal volume, including read and write capabilities. Edwards at 135. In other words, a writable clone or virtual copy of a database can be created based on a snapshot of the database created and stored in a remote destination. Zadok Decl. at ¶ 72.

Edwards further explains that “[c]reating a clone volume is a simple process.” Id. at 135. A container file for the new clone volume (or FlexClone volume) is created and seeded “with a vol_info block that is a copy of the vol_info block of the Snapshot copy on which the clone is based.” Id. Since the vol_info block is the root of the “tree of blocks that form the Snapshot copy, the clone inherits pointers to the complete file system image stored in the original Snapshot
copy.” *Id.* at 135. This process is shown below with respect to Edwards’s Figure 1, annotated:

![Diagram of WAFL data structures](image)

By creating a new vol_info block that points to the tree of blocks of the snapshot (point-in-time) copy, Edwards discloses creating a new, virtual file system comprising a set of files linked to the data blocks associated with the base snapshot, as shown above. Zadok Decl. at ¶ 74. This is the same way that the ‘361 patent creates files for a virtual database, by implementing them “as pointers to the actual database block that stores the data.” ‘361 patent at 19:47-49; Zadok Decl. at ¶ 74.

The created files in a FlexClone volume are capable of being read from and written to. *See* Edwards at 129 (FlexClone volumes are “writable Snapshot® copies”). And like all FlexVol volumes, FlexClone volumes are virtual in that they “separate the management of physical storage devices from the management of
logical data sets.” *Id.* at 141; Zadok Decl. at ¶ 75. As such, a database residing in a FlexClone volume is a virtual database, since it comprises a set of database files capable of being read from and written to, created by pointing to already-stored database blocks. *Id.*

**D. Chapman (Ex. 1006)**

Chapman, entitled “SnapMirror® Best Practices Guide,” is a NetApp, Inc. technical report (TR-3446) that was published in April 2006 (Chapman at 1) and revised and published in May 2006 (Chapman at 63), is prior art under § 102(b) and was not before the USPTO during prosecution of the ‘361 patent. Chapman is a guide to implementing SnapMirror. Zadok Decl. at ¶ 76.

Chapman discloses SnapMirror “cascading,” or transmitting data from a SnapMirror destination storage system to a second destination storage system, i.e., “from A to B to C and so on” using SnapMirror volume replication. Zadok Decl. at ¶ 77; Chapman at 22. “SnapMirror volume replication is a block-for-block replication; it transfers the file system verbatim.” *Id.* at 10. Data is transferred “from one destination to another in a series.” *Id.* at 11; Zadok Decl. at ¶ 77. The

---

6 The “cascading” feature of SnapMirror is relevant to the challenged claims which may be implemented using successive SnapMirror data transfers. Zadok Decl. at ¶ 77 and footnote 5.
following is an exemplary illustration of data on a source system (the leftmost box) cascaded to two successive destination systems (the middle and rightmost boxes) using Volume SnapMirror replication (VSM):

<table>
<thead>
<tr>
<th>CONFIGURATION OF SNAPMIRROR RELATIONSHIPS</th>
<th>SUPPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td>✓</td>
</tr>
</tbody>
</table>

*Id.* at 61 (other configurations displayed on page 61 omitted).

Chapman further discloses cloning any SnapMirror volumes, including SnapMirror volumes that reside in a cascaded SnapMirror destination volume:

*Users can clone SnapMirror volumes.* Users must be aware that a clone of a SnapMirror destination locks down the Snapshot copy from which the clone was created. In doing so, it also locks down that copy in the source volume, and *in every volume in the cascade, if the volume is part of a SnapMirror cascade.*

Chapman at 19. In other words, any volume that resides on a cascaded, SnapMirror destination storage system can be cloned. Zadok Decl. at ¶ 78. Chapman also discloses that a “[w]hen a tape device is attached to a Network Appliance SnapMirror destination, data can be moved to tape periodically.” *Id.* at 6.

Advantages to using SnapMirror for tape backup consolidation include “reducing backup administrative requirements at remote locations” and “dramatically reduc[ing] overhead from stressful backup operations ….” *Id.*

**VIII. MOTIVATIONS TO COMBINE THE NETAPP PRIOR ART REFERENCES**
A. Motivation To Combine References From The Same Company That Describe The Same Product

A POSITA would have been motivated to combine any of the NetApp prior art references (Sanders, Edwards, and Chapman). In *Ex parte Mettke*, the Board ruled in an appeal of a reissue application that there was “motivation to combine any of the teachings” of four prior art references to invalidate the claim because they are “from the same corporation, and expressly teach modifications, variations, and improvements to a pay-for-use public communications terminal.” No. 2008-0610, 2008 WL 4448201 (BPAI Sept. 30, 2008), (attached as Ex. 1017), aff’d, 570 F.3d 1356 (Fed. Cir. 2009). And because the four references “are all from the same corporation and all relate to versions of the same pay-for-use terminal,” the Board found that a POSITA “would have been motivated to combine the teachings in one reference with teaching in another reference because they are all related to the same terminal apparatus.” *Id.* The Board noted that in this situation, “there is not the usual obviousness problem of explaining why one skilled in the art would have sought to combine two references from unrelated sources.” *Id.* *Ex parte Mettke* is indistinguishable from this case and thus a POSITA would have been motivated to combine any of the NetApp prior art. See also Zadok Decl. at ¶¶ 81-84.

B. Motivation to Combine Sanders with Edwards and Chapman

(A) Same Problem & Same Solution:
Sanders, Edwards, and Chapman all address the same problem – creating virtual copies of databases to avoid the costs and complexities involved in making additional physical copies of the data for development, backup, recovery, data warehousing, and other purposes. See Zadok Decl. at ¶ 85. Sanders, Edwards, and Chapman all address this problem in the same manner. Id. For example, Sanders describes that “FlexClone technology … can be used to create an instantaneous point-in-time copy of a production database.” Sanders at 3; see also Edwards at 134 (“Creating a [] FlexClone volume . . . is nearly instantaneous”); Chapman at 19 (“Cloning offers a nearly instantaneous replica of a volume within the same aggregate.”).

Sanders, Edwards, and Chapman describe enhancing the read-only point-in-time copies that reside in a destination storage system with FlexClone technology for replication and disaster recovery purposes. Zadok Decl. at ¶ 86; see also Sanders at 5 (explaining that FlexClone technology can greatly enhance disaster recovery sites by creating a “[w]riteable Disaster Recovery Destination”); Edwards at 134 (“sometimes the read-only nature of a Snapshot copy is a limitation. In database environments, for example, it is often desirable to make writable copies of a production database for development or test purposes.”); Chapman at 19 (“It is also possible to create a writable volume from a read-only SnapMirror destination. […] Cloning does not require any special hardware.”).
Sanders, Edwards, and Chapman discuss the same solution to the problem—using snapshot copies for data replication, and to create virtual writable copies of a database for applications such as data warehousing. See Zadok Decl. at ¶ 87; see also Sanders at 3, 12; Edwards at 134; Chapman at 5-6. Sanders leverages snapshots to create space efficient virtual writable clones or copies. Zadok Decl. at ¶ 87. Indeed, Sanders discloses that “[t]he clone volume uses space very efficiently, allowing both the original FlexVol volume and the FlexClone volume to share common data, storing only the data that changes between the original volume and the clone.” Sanders at 3; see also Edwards at 129 (disclosing that clone volumes “allow the creation of light-weight copies of live data sets while consuming minimal storage resources.”); Chapman at 19 (disclosing that clones offer “substantial space savings for work environments that require multiple copies of the same data.”). A POSITA using Sanders’s method for cloning databases would have looked to Edwards and Chapman for further architectural explanation and deployment details related to the cloning methods described. Edwards at 129 (describing itself as presenting “the basic architecture of FlexVol” used to enhance SnapMirror and to create clones); Chapman at 5 (“This technical report is designed for storage administrators and architects who are already familiar with SnapMirror software and are considering deployments for production environments. […] The purpose of this paper is to present a guide for implementing SnapMirror
technology, addressing step-by-step configuration examples . . . ”).

Accordingly, a POSITA had every motivation to combine the teachings of Sanders with Edwards and Chapman and vice-versa. Zadok Decl. at ¶ 88.

(B) Nothing Incompatible:

Nothing in these prior references teaches away from their combination. Zadok Decl. at ¶ 88. There is no essential feature in any of these references that conflicts with an essential feature of any other reference. Id. To the contrary, these references are a natural fit, as evidenced by actually being combined by NetApp to replicate database data and to create virtual databases. Id. The FlexClone, FlexVol, and SnapMirror technologies described in Sanders, Edwards and Chapman were used to create virtual database systems and were offered together in NetApp’s Data ONTAP system well before the date of the earliest patent application that led to the ‘361 patent. Id. Thus, they were all compatible since they worked together on the same NetApp operating system. Id.

IX. PRECISE REASONS FOR THE RELIEF REQUESTED

The purported invention of the ‘361 patent is predicated upon the creation of a virtual file system that uses existing point-in-time copies of a source file system (“snapshots”) stored on a storage system. These technologies were well known in the art for years prior to the filing date. For the following precise reasons, the challenged claims of the ‘361 patent are invalid in view of the prior art.
A. **Ground 1: Claims 1-6, 8, 14, 16-19, 24, and 25 are rendered obvious over Sanders in view of Edwards and Chapman.**

1. **Claim 1**

[1a] “A method for replicating a database, the method comprising:”

The preamble of Claim 1 is not limiting at least because it is duplicative of the limitations in the claim’s body and does not recite any essential structure or steps. Regardless, as discussed in further detail below, Sanders discloses a process for replicating a database. Zadok Decl. at ¶ 114; Sanders at 3 (“Database cloning is [a] process by which you can create an exact copy of a DB2 database, either by physically copying the data or by performing what is known as a redirected restore.”). Accordingly Sanders discloses a method for replicating a database. Zadok Decl. at ¶ 115.

[1b] “linking a source database, wherein linking the source database comprises receiving information identifying the source database;”

Sanders discloses receiving information identifying a source database to be cloned. Zadok Decl. at ¶ 116. For example, Sanders discloses creating a snapshot copy of a source volume containing an IBM DB2 database on a source storage system and then mirroring the copy from the source volume to a destination volume on a first storage system. Id.; Sanders at 22-23. A command executed on the first storage system identifies the source volume on the source storage system:

```
snapmirror initialize -S [SourceStorageSystem]:[VolumeName][DestinationStorageSystem]:[Vol```
where

- **SourceStorageSystem identifies the name assigned to the SnapMirror source storage system.**

*Id.* at 22; Zadok Decl. at ¶ 116. The claim requires “linking” a source database, which comprises “receiving information identifying” a source database. Thus, Sanders discloses that the first storage system receives information identifying—and accordingly, “links”—the source database when the “snapmirror initialize” command is executed on the first storage system and provided with the source volume information – i.e., information identifying the source database. Zadok Decl. at ¶ 117.

[1c] “loading the source database at multiple points in time, wherein the loading comprises:

- receiving database blocks for different point-in-time copies of the source database, and
- storing the database blocks in a first storage system;”

Sanders discloses loading the source database at multiple points in time, comprising receiving database blocks for different point-in-time copies of the source database residing in a FlexVol volume, and storing those database blocks in a first destination storage system. Zadok Decl. at ¶ 118. For example, Sanders discloses that execution of the “snapmirror initialize” command on the first storage system described above results in the creation of a point-in-time (snapshot) copy of the source volume at the time the command is executed. *Id.; see* Sanders at 22
(“The SnapMirror initialize command creates a Snapshot copy of the source volume”). This point-in-time copy is stored on the destination volume in the first storage system as part of the initialization process triggered by the same command:

As noted earlier, the initialization process transfers data, including all Snapshot copies, from the source volume to the destination volume for the first time; thereafter, only changed blocks are transferred. Sanders at 21; Zadok Decl. at ¶ 118.

Sanders additionally discloses that after the SnapMirror initialization process has completed, a snapshot copy of the source volume is created on the source volume by executing a “snap create” command on the source storage system. Id. at ¶ 119. Then, Sanders discloses the execution of a “snapmirror update” on the first storage system “to update the destination volume with the contents of the source volume, including Snapshot copies.” Id. at ¶ 120; see also Sanders at 27. As a result, the destination volume in the first storage system receives and stores the snapshot created with the “snap create” command. Id. at 28; Zadok Decl. at ¶ 120. Accordingly, Sanders discloses receiving and storing different point-in-time copies of the source database in a FlexVol volume that are stored on the destination volume in a first destination storage system. Zadok Decl. at ¶ 121.

Moreover, Sanders in view of Edwards discloses that the received and stored point-in-time copies comprise “database blocks” – units of data used by a source database. Id. at ¶ 122. Specifically, Sanders discloses that each received snapshot
copy consists of “only changed blocks [that] are transferred.” Sanders at 21.

Edwards further discloses that the “data blocks” or “blocks” disclosed in Sanders and used by NetApp are units of data (4KB in NetApp’s WAFL file system) used by a database, which could be a source database. Id. at ¶ 122; see Edwards at 141 (describing the “fine grain of a file system (4KB in WAFL)”; id. at 135 (disclosing a source, FlexVol volume containing a “400 GB database table”).

Accordingly, Sanders in view of Edwards discloses receiving and storing database blocks for different point-in-time copies of a source database in a first storage system.

A POSITA would have known to combine the NetApp Data ONTAP technologies described in Sanders with the teachings of Edwards, which describes _________________

7 To the extent that the Board adopts a construction of a “database block” as requiring storage of both data and metadata associated with the database block, a POSITA would have understood that a FlexVol volume that stores a database, such as an IBM DB2 database, necessarily includes both data and metadata in each data block. Zadok Decl. at ¶ 122, footnote 8. A POSITA would know that databases running on a NetApp storage system result in NetApp data blocks that contain both the database’s data and metadata, such as “the database object [the] block belongs to.” ‘361 patent at 14:36-37; Zadok Decl. at ¶ 122, footnote 8.
in more detail “the WAFL file system, the core component of NetApp’s operating system, Data ONTAP,” in order to discern the units of data used by a database in WAFL. Id. at ¶ 124; Edwards at 130. Accordingly, Sanders in view of Edwards discloses this limitation. Id. at ¶ 125.

[1d] “replicating the database blocks of the source database from the first storage system to a second storage system; and”

Sanders discloses the use of NetApp’s SnapMirror technology for data replication. Zadok Decl. at ¶ 126; Sanders at 22, 27. Chapman discloses the use of data cascading, which is a “variation on the basic SnapMirror deployment and function involves mirroring from established mirrors to more SnapMirror destinations.” Chapman at 22; see Zadok Decl. at ¶ 126.

As discussed above, the first storage system is a SnapMirror destination. See supra [1c]. A SnapMirror cascade results in “copying from one destination to another in a series” using SnapMirror volume replication. Chapman at 11. Accordingly, a SnapMirror relationship is initialized between the first and second storage systems, an initial baseline snapshot copy of the source volume is transferred from the first system to a destination volume on the second system, and subsequently only changed blocks are transferred. Zadok Decl. at ¶ 127; Chapman at 10 (describing SnapMirror volume replication, including initial and changed-block transfers, under “VSM Detail”). “SnapMirror volume replication is a block-for-block replication; it transfers the file system verbatim.” Id. at 9. Thus,
Chapman discloses “replicating the database blocks of the source database from the first storage system to a second storage system.” Zadok Decl. at ¶ 127.

A POSITA would have been motivated to combine Sanders with Chapman to replicate the source database blocks from the first storage system to a second storage system because both references disclose the same SnapMirror technology to replicate data from databases to a remote destination, and Chapman explicitly discloses best practices for using SnapMirror, including data cascades for data distribution. Zadok Decl. at ¶ 128; Chapman at 22 (disclosing SnapMirror “Cascading”); see also §§ VIII.A, VIII.B. Given Sanders’s disclosure that a “SnapMirror source and its corresponding destination can reside on […] two separate storage systems that are miles apart” (Sanders at 3), a POSITA would have been motivated to incorporate Chapman’s teachings that cascading “make[s] a uniform set of data available on a read-only basis to users from various locations throughout a network” (Chapman at 22) and can thus be “very useful for data distribution.” Id. at 11. Thus, Sanders in view of Chapman discloses this limitation. Zadok Decl. at ¶ 129.

[1e(i)] “provisioning a virtual database (VDB) from the second storage system to a system running a database server, wherein provisioning comprises:”

The ’361 patent’s specification defines “provisioning” a virtual database as “[t]he process of making the virtual database available to a database server.” ’361
patent at 6:47-49. Sanders discloses making the virtual database available to a database server running on a storage system through the use of FlexClone to create clones of databases on FlexVol volumes, followed by the mounting of those clones to a database server. Zadok Decl. at ¶ 130; Sanders at 3 (“A database clone can be created simply by creating a clone of the FlexVol volumes that are used for the database’s data and transaction logs.”); see also id. at 28 (“In order to access the clone database, you need to mount the clone volumes to a database server”). Chapman further discloses cloning any SnapMirror volume, including SnapMirror volumes that reside in a cascaded SnapMirror destination volume such as the second storage system:

Users can clone SnapMirror volumes. Users must be aware that a clone of a SnapMirror destination locks down the Snapshot copy from which the clone was created. In doing so, it also locks down that copy in the source volume, and in every volume in the cascade, if the volume is part of a SnapMirror cascade.

Chapman at 19. In other words, any volume that resides on a cascaded, second destination storage system can be cloned. Zadok Decl. at ¶ 131. Edwards likewise emphasizes that a clone can be created from “any FlexVol volume, including a read-only mirror.” Id.; Edwards at 134.

As discussed above in connection with claim element [1d], a POSITA would have been motivated to combine Sanders with Chapman to incorporate Chapman’s
teaching that cascading was “very useful for data distribution” (Chapman at 11) to replicate the database blocks of the source database from the first destination storage system to the second destination storage system, and to create a clone on the second destination storage system. Zadok Decl. at ¶ 132; see also §§ VIII.A and VIII.B. Thus the combination of Sanders with Chapman discloses creating and mounting a FlexClone volume on the second storage system that points to stored data blocks for a source database volume. Zadok Decl. at ¶ 132.

As explained below with respect to claim element [1(e)(iii)], Sanders discloses that the FlexClone volume is provisioned to a system running a database server. Zadok Decl. at ¶ 133. Moreover, as explained below with respect to claim element [1(e)(ii)], a clone database created in Sanders is a “virtual database.” Id. The combination of Sanders with Chapman thus discloses this limitation. Id.

[1e(ii)] “creating a set of files linked to the stored database blocks on the second storage system, and”

Sanders discloses that “[a] database clone can be created simply by creating a clone of the FlexVol volumes that are used for the database’s data and transaction logs.” Sanders at 3. The resulting clone database “is a frozen image of the database file system at the time of the clone creation.” Id. at 6. Zadok Decl. at ¶ 135.

Edwards, a NetApp technical article, provides further detail on creating a set of files linked to the stored database blocks on the storage system, through the
creation of a FlexClone volume. Id. As discussed in detail previously in § VII.C, Edwards explains how a clone is created from a snapshot, including a snapshot stored on a destination storage system. Edwards explains that “[c]reating a clone volume is a simple process. WAFL creates the files required for a new FlexVol volume.” Edwards at 135. A container file for the new clone volume is created and seeded “with a vol_info block that is a copy of the vol_info block of the snapshot copy on which the clone is based.” Id. Thus, the cloned volume inherits all the pointers or links of the snapshot copy that point to the underlying data blocks. Effectively, the clone volume “inherits pointers to the complete file system image stored in the original Snapshot copy.” Id. By creating a new vol_info block that points or links to the tree of blocks of the snapshot (point-in-time) copy, Edwards discloses creating a new, cloned volume comprising a set of database files, by pointing to already-stored database blocks associated with a point-in-time copy of the source database. Zadok Decl. at ¶ 136.

Moreover, the FlexClone volume containing a virtual copy of a source database is a writable copy of a source database. Id. at ¶ 137; Edwards at 135 (a FlexClone volume—a virtual database—is “a full-fledged FlexVol volume with all the features and capabilities of a normal WAFL volume.”). Accordingly, a FlexClone volume containing a database is capable of being read and written to, just like any other FlexVol volume. Zadok Decl. at ¶ 137; Edwards at 129
(“**FlexClone**® volumes provide **writable** Snapshot® copies.”) (italics in original).

As such, Edwards discloses creating a virtual database: a set of database files capable of being read from and written to, created by pointing to already-stored database blocks. Zadok Decl. at ¶ 137.

To the extent the set of files created for a virtual database needs to “include[] a set of database blocks and the data structures for referring to the database blocks,” ‘361 patent at 6:34-35, this is also disclosed by Edwards. Zadok Decl. at ¶ 138; Edwards at 130 (explaining that each file “is described by an **inode**, which contains per-file metadata and **pointers to data or indirect blocks**”) (italics in original). The data blocks of the snapshot copy in Edwards are the data blocks of the virtual database’s files and the new vol_info block and its associated pointers are the data structures for referring to these data blocks. Zadok Decl. at ¶ 138.

Finally, the virtual database disclosed by Edwards is virtual in that the physical implementation of the database files is decoupled from their logical use. *Id.* at ¶ 139. FlexClone volumes “separate the management of physical storage devices from the management of logical data sets.” *Id.*; Edwards at 141.

A POSITA would have been motivated to combine Sanders and Edwards, given that they recite the same FlexClone, SnapMirror, and snapshot features from the Data ONTAP platform by NetApp, and a POSITA would have looked to Edwards to understand the specifics of how FlexClone creates files linked to stored
database blocks on the storage system. Zadok Decl. at ¶ 140. Accordingly, the combination of Sanders with Edwards discloses this limitation. *Id.*

[1e(iii)] “mounting the set of files to the system to allow a database server running on the system to access the set of files.”

Sanders discloses a command to mount cloned NetApp volumes: “[Y]ou can mount the clone volume by executing the following command on the database server: mount [MountPoint], where • *MountPoint* identifies the name assigned to the mount location that is used to mount the flexible volume on the database server.” Sanders at 29; see also *id.* at 9, Fig. 4, Step 5; Zadok Decl. at ¶ 141.

By identifying a mount location in the storage system, the set of files in the cloned volume can be made accessible to the system over a file sharing system (NFS), allowing a database server running on the system to access the clone volumes. *Id.* at ¶ 142; Sanders at 28 (”*In order to access* the clone database, you need to *mount* the clone volumes to a database server.”). Thus, Sanders discloses this limitation, and the combination of Sanders with Edwards and Chapman meets the limitations of Claim 1 and renders it obvious. Zadok Decl. at ¶ 142.

### 2. **Claim 2**

“2. The method of claim 1, wherein the replicating of database blocks comprises transmitting the database blocks from the first storage system to the second storage system for storing in the second storage system.”

As discussed above in connection with claim element [1d], the replicating of data from the first storage system to a second storage system is part of a
SnapMirror cascade, i.e., “copying from one destination to another in a series.” Chapman at 11; Zadok Decl. at ¶ 144. As such, “After performing an initial transfer of all data in the volume, VSM [Volume SnapMirror] sends to the destination only the blocks that have changed since the last successful replication.” Chapman at 10. Moreover SnapMirror volume replication is “block-for-block replication of layout of data on disk.” *Id.* Accordingly, Chapman discloses transmitting database blocks from a first storage system to a second storage system for storing on a storage system. Zadok Decl. at ¶ 144. Since Claim 2 is dependent on Claim 1, the combination of Sanders with Edwards and Chapman meets the limitations of Claim 2 and renders it obvious. *Id.*

3. **Claim 3**

“3. The method of claim 1, wherein the replicating of database blocks comprises transmitting a subset of database blocks comprising database blocks that changed since a point-in-time from the first storage system to the second storage system for storing in the second storage system.”

As discussed in connection with claim element [1d], the replicating of data from the first storage system to a second storage system is part of a SnapMirror cascade replicating the data in a volume, i.e., “copying from one destination to another in a series.” Chapman at 11; Zadok Decl. at ¶ 145. Accordingly, a SnapMirror relationship exists between the first and second storage systems. *Id.*

As discussed above connection with Claim 2, Chapman discloses that after an initial baseline transfer of all of the data in a source volume, SnapMirror “sends
to the destination **only the blocks that have changed** since the last successful replication.” Chapman at 10. The last successful replication is a particular point-in-time, and the “blocks that have changed” on the source volume since that time comprise a subset of all of the blocks stored on the volume. A POSITA using Sanders to clone databases would have looked to Chapman for information on SnapMirror for the reasons described in claim element [1d]. Zadok Decl. at ¶ 146. Accordingly, Chapman discloses the limitations of Claim 3, and since Claim 3 is dependent on Claim 1, the combination of Sanders with Edwards and Chapman meets the limitations of Claim 3 and renders it obvious. *Id.*

4. **Claim 4**

“4. The method of claim 1, further comprising: using the virtual database from the second storage system as a standby database for use when the source database is unavailable.”

As described above, a standby database means “a database (including a virtual database) that may be used in lieu of another database that is unavailable.” *See* § V.B.4. As discussed above in connection with claim element [1e(i)], the “virtual database” resides within the FlexClone volume on the second storage system that replicates the source database volume.

Sanders discloses using the virtual database from the second storage system as a standby database when the source database is unavailable. Sanders discloses that users may use a clone database as a “Standby Database” to “[i]mmediately
resume read-write workload on discovering corruption in the production dataset by mounting the clone instead.” Sanders at 5. One of the reasons that a production database may be unavailable is corruption of its data. Zadok Decl. at ¶ 147. Accordingly, Sanders discloses the limitations of this claim and since Claim 4 is dependent on Claim 1, the combination of Sanders with Edwards and Chapman meets the limitations of Claim 4 and renders it obvious. Id. at ¶ 148.

5. **Claim 5**

“5. The method of claim 1, further comprising: transmitting database blocks stored in the second storage system to the first storage system; and storing the database blocks received from the second storage system at the first storage system.”

As discussed in Claim 1, a SnapMirror relationship is initialized between FlexVol volumes on a first storage system and a second storage system, and a clone database created from the replicated data on the volume in the second storage system. See supra Claim 1. The ’361 patent discloses that use of a standby database when a source database is unavailable requires the resynchronization of its data with the source database when the source database is ready to resume operations. See ‘361 patent, 30:11-17. Chapman discloses resynchronization of a SnapMirror relationship. Zadok Decl. at ¶ 150; Chapman at 18 (“Resyncing Destination to Source When Destination Is FlexVol Capable”).

In particular, the “snapmirror resync” command can be applied to resynchronize the SnapMirror source volume with the destination volume, by
transmitting all or a subset of blocks from the destination to the source:

Resync does not always require a full level 0 transfer. SnapMirror looks at the most recent common snapshot between the source and destination and will resync only the changed blocks since the most recent common snapshot. In the absence of any common snapshots, resync will require a full transfer.

*Id.*; Zadok Decl. at ¶ 150. Given the disclosure of a SnapMirror relationship between the first and second storage systems and the “snapmirror resync” command to reverse the role of the source and destination so that database blocks stored in the second storage system are transmitted to and stored on the first storage system, Chapman discloses this limitation. Zadok Decl. at ¶ 151. A POSITA would have been motivated to combine this disclosure with Sanders and Edwards because Chapman explains additional options and best practices for using the SnapMirror tool expressly discussed in Sanders and Edwards. *Id.* Specifically, the SnapMirror resync command is used when “the source needs to match” the destination, i.e., for recovering data that has been replicated or backed up on the destination FlexVol volume. Chapman at 18. 8 This scenario is consistent with

8 Chapman notes that “snapmirror resync” cannot be used with a SnapMirror destination that supports a FlexClone volume. Chapman at 19. However, Chapman teaches that “it is a good idea to split clones that have been used for an extended
SnapMirror’s main purpose as taught in Sanders: to recover data from a remote site for disaster recovery. Zadok Decl. at ¶ 151; see also § VII.B.

Accordingly, Chapman discloses the limitations of Claim 5, and since Claim 5 is dependent on Claim 1, the combination of Sanders with Edwards and Chapman renders Claim 5 obvious. Zadok Decl. at ¶ 152.

6. **Claim 6**

“6. The method of claim 1, further comprising: receiving request to update database blocks associated with the VDB from the database server running on the system.”

Sanders discloses that “a FlexClone volume is a writable point-in-time image of a FlexVol volume or another FlexClone volume.” Sanders at 3. Sanders discloses mounting the FlexClone volume for access by a database server, wherein options are defined including read/write (“rw”) permissions.

For example, for a clone volume named dbdata_cl that resides on a storage system named srcstore, you would append the following entry to the /etc/fstab file on the database server:

```
srcstore:dbdata_cl /mnt/dbdata_cl nfs
hard, rw, nointr, rsize=32768, wsize=32768, bg, vers=3, tcp 0 0
```

period of time,” such as a clone used as a standby database. *Id.* Splitting the FlexClone volume enables resynchronization by “remov[ing] the connection between the clone and its parent.” *Id.*; Zadok Decl. at ¶ 151 and footnote 9.
Id. at 29. The “rw” notation specifies that the clone database is readable and writable. See also id. at 5 (“both reads and writes are allowed on a cloned database’’); Zadok Decl. at ¶ 153. Thus the FlexClone is writable by the database server. Id. In order for the database server to write to a database a write request must first be received from the database server, and the clone database’s data blocks will be accordingly updated. Id. Delphix’s infringement contentions for Claim 6 is consistent with the writable functionality of a clone database meeting this limitation: Delphix merely points to the Accused Products’ “creat[ing] virtual copies of data blocks obtained from production databases and mak[ing] virtualized copies available to other servers for both reading and writing . . . .” Ex. 1008 at 26.

Accordingly Sanders renders the limitations of this claim obvious, and since Claim 6 is dependent on Claim 1, the combination of Sanders with Edwards and Chapman renders this claim obvious. Zadok Decl. at ¶ 155.

7. **Claim 8**

[8a(i)] “8. The method of claim 1, further comprising: provisioning a second virtual database from the first storage system to a second system running a second database server, wherein provisioning comprises:”

The ‘361 patent’s specification defines “provisioning” a virtual database as “the process of making the virtual database available to a database server.” ‘361 patent at 6:47-49. As discussed in connection with claim element [1e(i)], Sanders discloses making a virtual database available to a database server through the use
of FlexClone to create a clone of a database on a FlexVol volume, followed by the mounting of the clone to a database server. Zadok Decl. at ¶ 156.

A POSITA would have understood that multiple clones can be created and mounted and would have been motivated to do so, given the many benefits of creating multiple clones for different purposes. Zadok Decl. at ¶ 157; Sander at 5 (various “key business requirements” that can benefit greatly from a clone database created using FlexClone technology” include, for example, disaster recovery and application upgrades). Moreover, Edwards clearly discloses creating multiple clones. Edwards at 134 (“it is often desirable to make writable copies of a production database for development or test purposes”); Zadok Decl. at ¶ 157. A POSITA also would have known to use NetApp’s FlexClone technology to create multiple clones from a production database, as described in both Sanders and Edwards. Id. at ¶ 158.

As previously discussed in connection with claim element [1e(i)] Chapman discloses cloning any SnapMirror volume, including SnapMirror volumes that reside in a cascaded SnapMirror destination volume such as the SnapMirror destination on the first storage system that receives database blocks for different point-in-time copies of the source database. Zadok Decl. at ¶ 159; Chapman at 19. A POSITA would have been motivated to combine Sanders’s disclosure of creating and mounting a clone database with the disclosure of Chapman that the
destination volume on the first storage system could be cloned in order to obtain the benefits of multiple clones. Zadok Decl. at ¶ 159. Moreover, as explained above in connection with claim element [1(e)(ii)], a clone database created in Sanders is a “virtual database.” Id. at ¶ 160. Accordingly the combination of Sanders with Edwards and Chapman discloses this limitation. Id.

[8a(ii)] “creating a second set of files linked to the stored database blocks on the first storage system, and”

As discussed previously in connection with claim element [1(e)(ii)], the combination of Sanders with Edwards discloses creating a set of files linked to storage blocks on a storage system. See [1e(ii)]; Zadok Decl. at ¶ 161. A POSITA would also have been motivated to combine Sanders with Edwards for the same reasons disclosed above in connection with claim element [1e(ii)]. Id. at ¶ 163. The combination of Sanders with Edwards thus discloses this limitation. Id.

[8a(ii)] “mounting the set of files to the second system allowing the second database server to access the set of files.”

As discussed previously in connection with claim element [1e(iii)], Sanders discloses mounting a set of files to a system to allow a database server running on the system to access the set of files. See also Zadok Decl. at ¶ 164. Accordingly, Sanders discloses this limitation, and the combination of Sanders with Edwards and Chapman meets the limitations of Claim 8 and renders it obvious. Id. at ¶ 165.

8. **Claim 14**

The preamble of Claim 14 is not limiting at least because it is duplicative of the limitations in the claim’s body and does not recite any essential structure or steps. Regardless, as discussed in further detail below, Sanders discloses a method for backup of source databases using a virtual database system. Zadok Decl. at ¶ 167; Sanders at 5 (“[K]ey business requirements that can benefit greatly from a clone database created using FlexClone technology: [include] … Online Database Backup: […] If necessary, the primary database can be restored from the snapshot created for the clone; or applications can point directly to the clone database.”). As explained previously in connection with claim element [1e(i)], the clone database volume is “virtual” because it “points” to the pre-existing data blocks associated with a point-in-time copy of its parent volume (containing the source database) instead of physically duplicating the data blocks of its parent. Zadok Decl. at ¶ 168. Accordingly Sanders in view of Edwards discloses a method for backup of source databases using a virtual database system. Id.

[14b] “linking a source database, wherein linking the source database comprises receiving information identifying the source database;”

Claim element [14b] is identical to claim element [1b]. Accordingly, for the same reasons described above with respect to claim element [1b], Sanders discloses this limitation. See § IX.A.1, claim element [1b]; Zadok Decl. at ¶ 170.
[14c] “loading a plurality of point-in-time copies of the source database, wherein the loading comprises: receiving database blocks for the point-in-time copies of the source database, and storing the database blocks on a first storage system; and”

Claim element [14c] is substantively identical to claim element [1c] and uses the same language except that it substitutes the phrase “loading a plurality of point-in-time copies of the source database … receiving database blocks for the point-in-time copies” for “loading the source database at multiple points in time … receiving database blocks for the different point-in-time copies.” For the same reasons described above with respect to claim element [1c], Sanders in view of Edwards discloses this limitation. See Sanders at 21 (describing transfer of snapshots from a source volume to a destination storage comprising initially “all Snapshot copies” and subsequently updating those snapshot copies, the subsequent transfers comprising “only changed blocks”); Edwards at 141, 135 (describing each block as a unit of data of 4 KB and describing a source volume comprising a database); § IX.A.1, claim element [1c]; Zadok Decl. at ¶ 171.

[14d(i)] “provisioning a virtual database to a system, wherein provisioning the virtual databases [sic] to a system comprises:”

As discussed above in connection with claim element [1e(i)], Sanders in view of Chapman discloses “provisioning a virtual database … to a system” through the use of FlexClone to create clones of databases on FlexVol volumes and the mounting of those clones to a database server running on the storage system.
See [1e(i)]; Zadok Decl. at ¶ 172. A POSITA would also have been motivated to combine Sanders with Chapman and provision a virtual database to a system to obtain the online backup and disaster recovery benefits disclosed in Sanders. Zadok Decl. at ¶ 173; Sanders at 6 (disclosing that “key business requirements that can benefit greatly from a clone database created using FlexClone technology” include “Writeable Disaster Recovery Destination,” and “Online Database Backup … If necessary, the primary database can be restored from the snapshot created for the clone …”). Accordingly, for the same reasons described above with respect to claim element [1e(i)], the combination of Sanders with Chapman discloses this limitation. Zadok Decl. at ¶ 173.

\[14d(ii)\] “creating a set of files linked to the stored database blocks on the storage system, and”

This limitation is identical to claim element [1e(ii)], except that it omits the word “second” before “storage system, and.” Zadok Decl. at ¶ 174. As discussed above in connection with claim element [1e(ii)], the combination of Sanders with Edwards discloses “creating a set of files linked to the stored database blocks on the storage system” by creating a FlexClone volume which is linked to data blocks of a point-in-time copy of a database stored on the storage system, where the volume contains a set of database files capable of being read from and written to, created by pointing to already-stored database blocks. See id.; see also [1e(ii)].

A POSITA would have been motivated to combine Sanders and Edwards to
create the “set of files,” given that both references recite the same FlexClone, SnapMirror, and snapshot features from the Data ONTAP platform by NetApp, and a POSITA would have looked to Edwards to understand how FlexClone creates files linked to stored database blocks on the storage system. Zadok Decl. at ¶ 175. Accordingly, for the same reasons described above with respect to claim element [1e(ii)], the combination of Sanders with Edwards discloses this limitation. See [1e(ii)]; Zadok Decl. at ¶ 175.

[14d(iii)] “mounting the set of files to the system to allow a database server running on the system to access the set of files”

This claim element is identical to claim element [1e(iii)] except that it uses the phrase “to allow a database server” in place of “allowing the database server.” Zadok Decl. at ¶ 176. Accordingly, for the same reasons described with respect to claim element [1e(iii)], Sanders discloses this limitation. Id.

[14e] “performing backup of the database blocks stored on the first storage system, wherein performing backup comprises transmitting database blocks associated with the source database from the first storage system to a second storage system.”

This claim element is substantially similar to claim element [1d], with an additional requirement that database blocks are transmitted or replicated for the purposes of backup. Whereas claim element [1d] requires “replicating the database blocks of the source database from the first storage system to a second storage system,” claim element [14e] requires “performing backup … wherein performing
backup comprises transmitting database blocks associated with the source database from the first storage system to a second storage system.”

For the same reasons described with respect to claim element [1d], Sanders in view of Chapman discloses replicating or transmitting snapshots comprising data blocks associated with a source database from a first SnapMirror destination to a second SnapMirror destination. See [1d]; Zadok Decl. at ¶ 178. Sanders in view of Chapman also discloses replicating or transmitting snapshots associated with a source database from a first SnapMirror destination to a second SnapMirror destination to perform backup. Id.; Sanders at 6 (“Online Database Backup: […] If necessary, the primary database can be restored from the snapshot created for the clone ….”); Chapman at 6 (“SnapMirror can also be used for backup consolidation and for offloading tape backup overhead from production servers. This facilitates centralized backup operations, reducing backup administrative requirements at remote locations.”). A POSITA would have been motivated to combine Sanders with Chapman for the same reasons described above with respect to claim element [1d] and also to obtain these backup-related benefits. Zadok Decl. at ¶ 179.

Accordingly Sanders in view of Chapman discloses this limitation, and the combination of Sanders with Edwards and Chapman meets the limitations of Claim 14 and renders it obvious. Id. at ¶ 180.

9. **Claim 16**
[16a] “16. The method of claim 14, wherein the source database is a first source database further comprising:”

[16b] “linking a second source database, wherein linking the second source database comprises receiving information identifying the second source database;”

[16c] “loading a plurality of point-in-time copies of the second source database comprising:

   receiving database blocks for the point-in-time copies of the second source database, and

   storing the database blocks on the first storage system; and”

Claim 14 claims a “method for backup of source databases using a virtual database system” and includes elements [14b] and [14c] relating to linking and loading a plurality of point-in-time copies of a “source database” on a “first storage system.” See ‘361 patent, Claim 14. Claim 16 depends on Claim 14, and claim elements [16b] and [16c] repeat the limitations of elements [14b] and [14c] with respect to a “second source database.” As discussed above in connection with Claim 14, the combination of Sanders with Edwards discloses the limitations of claim elements [14b] and [14c]. Zadok Decl. at ¶ 182.

The method for backup of source databases using a virtual database system disclosed by Sanders in view of Edwards and Chapman in Claim 14 is not limited to a single source database. Id. at ¶ 183. Indeed the method may simply be repeatedly applied to back up multiple databases to the same storage system, and a POSITA would have been motivated to do so in order to gain the cumulative
benefits realized from its repeated application to multiple databases containing
critical data. *Id.*; Chapman at 17 (disclosing that “[i]f critical data is mirrored to a
different physical location, a serious disaster does not necessarily mean extended
periods of data unavailability”). In particular, a POSITA would have known to
repeat the method of Claim 14 to backup a “second source database” to obtain
those benefits. Zadok Decl. at ¶ 183.

SnapMirror replication is “volume-based” (Chapman at 4), i.e., volume-to-
volume and not system-to-system; accordingly, the receiving and storing of the
database blocks associated with the second database on the first storage system
would not conflict with the receiving and storing of the database blocks associated
with the first database on that same system. Indeed, Chapman discloses that a
storage system may have multiple replication relationships. *Id.* at ¶ 184; Chapman
at 17 (disclosing that up to 600 “entries for each individual storage system” may be
specified in the file defining SnapMirror relationships). Accordingly, for these
reasons and the reasons discussed above in connection with Claim 14, the
combination of Sanders with Chapman discloses linking a second source database
and loading a plurality of point-in-time copies of the second source database on the
first storage system. *Id.* at ¶ 185.

[16d] “performing backup of the database blocks stored on the first
storage system, wherein performing backup comprises transmitting
database blocks associated with the second source database from the first
storage system to a second storage system.”

As discussed immediately above, a POSITA would have known to apply the method of Claim 14 a second time to back up a “second source database.” Zadok Decl. at ¶ 186. For the reasons discussed above in connection with claim element [14e], the combination of Sanders and Chapman discloses backing up the database blocks associated with the second source database by using SnapMirror technology to replicate the source volume containing those blocks on the first storage system onto a destination volume on a second storage system. Id. Accordingly, the combination of Sanders with Edwards and Chapman meets the limitations of Claim 16 and renders it obvious. Id.

10. Claim 17

“17. A computer program product having a non-transitory computer-readable storage medium storing computer-executable code for replicating a database, the code comprising instructions to:

link a source database, wherein linking the source database comprises receiving information identifying the source database;

load the source database at multiple points in time, wherein instructions to load comprise instructions to:

receive database blocks for different point-in-time copies of the source database, and

store the database blocks in a first storage system;

replicate the database blocks of the source database from the first storage system to a second storage system; and
provision a virtual database (VDB) from the second storage system to a system running a database server, wherein instructions to provision comprise instructions to:

create a set of files linked to the stored database blocks on the second storage system, and

mount the set of files to the system allowing the database server running on the system to access the set of files.”

Claim 17 claims “computer-executable code … comprising instructions to” perform steps essentially identical to those recited in Claim 1. Zadok Decl. at ¶ 187. That the steps of Claim 17 are couched as “a computer program product having […] code comprising instructions to” perform the steps of Claim 1 does not materially distinguish Claim 17 from Claim 1 in any way with respect to invalidity analysis. See id. at ¶ 188. NetApp’s snapshot, SnapMirror, FlexVol, and FlexClone feature described in Sanders, Edwards, and Chapman, are computer features implemented in code loaded onto a physical medium programmed to perform each of the stated functions of Claim 1. See id.; Sanders at 6 (“It is assumed that the NetApp FAS or IBM N Series storage systems used are loaded with Data ONTAP 7.0 or later”). Thus, for the reasons discussed above in connection with Claim 1, Sanders, in view of Edwards and Chapman, discloses and renders obvious the computer program product elements of Claim 17. The motivations to combine Sanders with Edwards, and Chapman are the same as explained above in connection with Claim 1. Zadok Decl. at ¶ 191. Accordingly, the combination of
Sanders with Edwards and Chapman renders Claim 17 obvious. *Id.* at ¶ 192.

11. **Claim 18**

“18. The computer program product of claim 17, wherein the replicating of database blocks comprises transmitting the database blocks from the first storage system to the second storage system for storing in the second storage system.”

As explained above in connection with Claim 17, Claim 17 merely repeats the limitations of Claim 1 in a computer program product format. Claim 2 depends on Claim 1 and adds a limitation relating to “transmitting the database blocks from the first storage system to the second storage system for storing in the second storage system.” Claim 18 similarly depends on Claim 17 and adds the same limitation as Claim 2. *See* Zadok Decl. at ¶ 193. Accordingly, for the reasons discussed above in connection with Claim 2, Chapman discloses the limitations of this claim, and since Claim 18 is dependent on Claim 17, for the reasons discussed above in connection with Claim 17, the combination of Sanders with Edwards and Chapman meets these limitations and renders Claim 18 obvious. *Id.* at ¶ 194.

12. **Claim 19**

19. The computer program product of claim 17, wherein the code further comprises instructions to:

transmit database blocks stored in the second storage system to the first storage system; and

store the database blocks received from the second storage system at the first storage system.

As explained above in connection with Claim 17, Claim 17 merely repeats
the limitations of Claim 1 in a computer program product format. Claim 5 depends on Claim 1 and adds two limitations relating to “transmitting database blocks” and “storing the database blocks.” Claim 19 similarly depends on Claim 17 and adds two “transmit database blocks” and “store the database block” limitations that are essentially identical to those of Claim 5, except that they are in a different tense. See Zadok Decl. at ¶ 195. Accordingly, for the reasons discussed above in connection with Claim 5, Chapman discloses the limitations of this claim, and since Claim 19 is dependent on Claim 17, for the reasons discussed above in connection with Claim 17, the combination of Sanders with Edwards and Chapman meets these limitations and renders Claim 19 obvious. Id. at ¶ 196.

13. **Claim 24**

24. A computer program product having a non-transitory computer-readable storage medium storing computer-executable code for performing backup of source databases using a virtual database system, the code comprising instructions to:

link a source database, wherein linking the source database comprises receiving information identifying the source database;

load a plurality of point-in-time copies of the source database, wherein the loading comprises:

receiving database blocks for the point-in-time copies of the source database, and

storing the database blocks on a first storage system; and

provision a virtual database to a system, wherein instructions to provision the virtual databases to a system comprise instructions to:

create a set of files linked to the stored database blocks on the storage system, and
mount the set of files to the system to allow a database server running on the system to access the set of files

perform backup of the database blocks stored on the first storage system, wherein performing backup comprises transmitting database blocks associated with the source database from the first storage system to a second storage system.

Claim 24 repurposes all of the steps of method Claim 14, written in the present continuous tense, into a “computer program product” claim, written in the present tense. That the steps of Claim 24 are couched as “a computer program product having […] code comprising instructions to” perform the steps of Claim 14 does not materially distinguish Claim 24 from Claim 14 in any way with respect to invalidity analysis. See Zadok Decl. at ¶¶ 197-198. NetApp’s snapshot, SnapMirror, FlexVol, and FlexClone features described in Sanders, Edwards, and Chapman are computer features implemented in code loaded onto a physical medium programmed to perform each of the stated functions of Claim 14. See id.; Sanders at 6 (“It is assumed that the NetApp FAS or IBM N Series storage systems used are loaded with Data ONTAP 7.0 or later”).

Thus, for the reasons discussed above in connection with Claim 14, Sanders, in view of Edwards and Chapman, discloses and renders obvious the computer program product elements of Claim 24. Zadok Decl. at ¶ 200. The motivations to combine Sanders, Edwards, and Chapman do not change because Claim 24 is substantively identical to Claim 14. Id. at ¶ 201. Accordingly, the combination of Sanders with Edwards and Chapman renders Claim 24 obvious. Id. at ¶ 202.
14. **Claim 25**

25. The computer program product of claim 24, wherein the source database is a first source database, wherein the code further comprises instructions to:

- link a second source database, wherein linking the second source database comprises receiving information identifying the second source database;
- load a plurality of point-in-time copies of the second source database comprising:
  - receiving database blocks for the point-in-time copies of the second source database, and
  - storing the database blocks on the first storage system; and
- perform backup of the database blocks stored on the first storage system, wherein performing backup comprises transmitting database blocks associated with the second source database from the first storage system to a second storage system.

As explained above in connection with Claim 24, Claim 24 merely repeats the limitations of Claim 14 in a computer program product format. Claim 16 depends on Claim 14 and adds limitations related to a second source database. Zadok Decl. at ¶ 203. Claim 25 similarly depends on Claim 24 and adds limitations that are essentially identical to those of Claim 16; the only difference is that the limitations of Claim 25 are written in the present tense and those of Claim 16 in the present continuous tense. *Id.* Accordingly, for the reasons discussed above in connection with Claim 16, the combination of Sanders with Chapman discloses the limitations of this claim, and since Claim 25 is dependent on Claim 24, for the reasons discussed above in connection with Claim 24, the combination of Sanders with Edwards and Chapman meets these limitations and renders Claim 25 obvious.
Id. at ¶ 204.

X. CONCLUSION

For the reasons set forth above, inter partes review of claims 1-6, 8, 14, 16-19, 24, and 25 of the ‘361 patent is requested.

Respectfully submitted,

Dated: October 21, 2014

By: /s/ Robert Steinberg

Robert Steinberg
Reg. No. 33144
Latham & Watkins LLP
355 South Grand Avenue
Los Angeles, CA 90071-1560
213.485.1234; 213.891.8763 (Fax)

Jonathan Link
Reg. No. 41548
Latham & Watkins LLP
555 Eleventh St., N.W., Suite 1000
Washington, DC 20004
202-637-2243; 202-637-2201 (Fax)

Counsel for Petitioner
CERTIFICATE OF SERVICE

The undersigned certifies that a complete copy of this Petition for *Inter Partes* Review of U.S. Patent No. 8,566,361 and all Exhibits and other documents filed together with this Petition were served on the official correspondence address for U.S. Patent No. 8,566,361 shown in PAIR and Delphix Corp.’s current patent counsel:

J. David Hadden  
FENWICK & WEST LLP  
Silicon Valley Center  
801 California Street  
Mountain View, CA 94041  
Telephone: 650.988.8500  
Facsimile: 650.938.5200

via FED-EX overnight delivery, on October 21, 2014

By: /s/ Robert Steinberg

Robert Steinberg  
Reg. No. 33144  
Latham & Watkins LLP  
355 South Grand Avenue  
Los Angeles, CA 90071-1560  
213.485.1234  
213.891.8763 (Fax)

Jonathan Link  
Reg. No. 41548  
Latham & Watkins LLP  
555 Eleventh Street, NW  
Suite 1000  
Washington, D.C. 20004-1304  
202.637.2200  
202.637.2201 (Fax)  
Counsel for Petitioner