

1 Jason G. Sheasby (SBN 205455)
jsheasby@irell.com
2 Andrew Strabone (SBN 301659)
astrabone@irell.com
3 Lucas S. Oxenford (SBN 328152)
loxenford@irell.com
4 Steven Marshack (SBN 364249)
smarshack@irell.com
5 IRELL & MANELLA LLP
1800 Avenue of the Stars, Ste. 900
6 Los Angeles, CA 90067
310-277-1010

7
8 Anthony Q. Rowles (SBN 301209)
trowles@irell.com
9 IRELL & MANELLA LLP
840 Newport Center Dr #400
Newport Beach, CA 92660
10 949-760-0991

11 *Counsel for Defendants and Counter-Plaintiffs*
12 *Valtrus Innovations Limited and Key*
Patent Innovations Limited

13
14
15 UNITED STATES DISTRICT COURT
16 NORTHERN DISTRICT OF CALIFORNIA
17 SAN FRANCISCO DIVISION

18 GOOGLE LLC,
19 Plaintiff and Counter-Defendant,
20 v.
21 VALTRUS INNOVATIONS LIMITED and
22 KEY PATENT INNOVATIONS LIMITED.,
23 Defendants and Counter-Plaintiffs.

Case No. 3:26-cv-00363-TLT

**DEFENDANTS AND COUNTER-
PLAINTIFFS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT FOR
DECLARATORY JUDGMENT**

24
25 Defendants and Counter-Plaintiffs Valtrus Innovations Ltd. ("Valtrus") and Key Patent
26 Innovations Ltd. ("Key Patent") (collectively, "Defendants") respectfully submit this Answer,
27 Affirmative Defenses, and Counterclaims in response to Plaintiff Google LLC's ("Google")
28 Complaint for Declaratory Judgment.

IRELL & MANELLA LLP
1800 Avenue of the Stars, Ste. 900
Los Angeles, CA 90067
310-277-1010

Case No. 3:26-cv-00363-TLT

DEFENDANTS' ANSWER

Defendants, by and through their undersigned counsel, hereby respond to Plaintiff's Complaint for Declaratory Judgment of Non-Infringement of U.S. Patent Nos. 8,379,538; 7,251,588; 8,370,416; 7,640,332; 7,904,686; 7,730,218; 7,936,738; 7,376,953 and 6,889,244 (Dkt. 1) as follows:

1. Defendants admit that Google purports to seek the relief in Paragraph 1.

2. Defendants are without information regarding who Google considers to be a "customer," and Google has not identified any specific customers or communications, and on that basis cannot admit allegations relating to Google's customers. Defendants otherwise deny the allegations of Paragraph 2.

3. Paragraph 3 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 3, except as they relate to the Asserted Patents included in Defendants' Counterclaim, *infra*.

4. Defendants admit that Valtrus has represented that it is "successor in interest to a substantial patent portfolio created by Hewlett Packard Enterprise and its predecessor companies, including Compaq, Verity, and Hewlett-Packard Development Company" and that Valtrus' business involves licensing the patents it acquired from HPE. To the extent the complaint references specific allegations in separate actions, the pleadings of any such actions speak for themselves, and therefore no response is needed. Except as expressly admitted, Defendants deny the allegations of this paragraph.

5. Defendants are without information regarding who Google considers to be a "customer," and Google has not identified any specific customers or communications, and on that basis cannot admit allegations relating to Google's customers. Defendants otherwise deny the allegations of Paragraph 5.

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

1 6. Defendants are without information regarding who Google considers to be a
2 “customer,” and Google has not identified any specific customers or communications, and on that
3 basis cannot admit allegations relating to Google’s customers. Defendants otherwise deny the
4 allegations of Paragraph 6.

5 7. Defendants are without information regarding who Google considers to be a
6 “customer,” and Google has not identified any specific customers or communications, and on that
7 basis cannot admit allegations relating to Google’s customers. Defendants otherwise deny the
8 allegations of Paragraph 7.

9 8. Defendants admit that Valtrus has sent licensing communications to Google, in
10 which Valtrus provided Google notice that Google needed a license to certain patents in the Valtrus
11 patent portfolio. Defendants otherwise deny the allegations of Paragraph 8.

12 9. Defendants admit that Google has been on notice of its need for a license to the
13 Valtrus patent portfolio since April 14, 2021. Valtrus has directed licensing communications to
14 Google, including the letter referenced, which speak for themselves. Defendants otherwise deny the
15 allegations of Paragraph 9.

16 10. Defendants admit that they had licensing communications with Google, which
17 speaks for itself. Defendants otherwise deny the allegations of Paragraph 10 and/or are without
18 information or knowledge sufficient to form a belief as to the allegations of Paragraph 10 and
19 accordingly deny the allegations of Paragraph 10.

20 11. Defendants admit that they sent the letter referenced to Google, which speaks for
21 itself. Defendants otherwise deny the allegations of Paragraph 11.

22 12. Defendants admit that they had licensing communications with Google, which speak
23 for themselves. Defendants otherwise deny the allegations of Paragraph 12.

24
25
26
27
28
DEFENDANTS’ ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF’S COMPLAINT

1 13. Defendants admit that they sent the email referenced, which speaks for itself.
2 Defendants otherwise deny the allegations of Paragraph 13.

3 14. Defendants admit that they sent the letter and email referenced, which speak for
4 themselves. Defendants otherwise deny the allegations of Paragraph 14.

5 15. Defendants admit that they sent the letter referenced, which speaks for itself.
6 Defendants otherwise deny the allegations of Paragraph 15.

7 16. To the extent the complaint references specific allegations in separate actions, the
8 pleadings of any such actions speak for themselves, and therefore no response is needed. Except as
9 expressly admitted, Defendants deny the allegations of this paragraph.
10

11 17. To the extent the complaint references specific allegations in separate actions, the
12 pleadings of any such actions speak for themselves, and therefore no response is needed. Except as
13 expressly admitted, Defendants deny the allegations of this paragraph.
14

15 18. To the extent the complaint references specific allegations in separate actions, the
16 pleadings of any such actions speak for themselves, and therefore no response is needed. Except as
17 expressly admitted, Defendants deny the allegations of this paragraph.

18 19. To the extent the complaint references specific allegations in separate actions, the
19 pleadings of any such actions speak for themselves, and therefore no response is needed. Except as
20 expressly admitted, Defendants deny the allegations of this paragraph.
21

22 20. To the extent the complaint references specific allegations in separate actions, the
23 pleadings of any such actions speak for themselves, and therefore no response is needed. Except as
24 expressly admitted, Defendants deny the allegations of this paragraph.

25 21. To the extent the complaint references specific allegations in separate actions, the
26 pleadings of any such actions speak for themselves, and therefore no response is needed. Except as
27 expressly admitted, Defendants deny the allegations of this paragraph.
28

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

22. To the extent the complaint references specific allegations in separate actions, the pleadings of any such actions speak for themselves, and therefore no response is needed. Except as expressly admitted, Defendants deny the allegations of this paragraph.

23. To the extent the complaint references specific allegations in separate actions, the pleadings of any such actions speak for themselves, and therefore no response is needed. Except as expressly admitted, Defendants deny the allegations of this paragraph.

24. Defendants are without information or knowledge sufficient to form a belief as to the allegations of Paragraph 24 and accordingly deny the allegations of Paragraph 24.

25. Defendants are without information or knowledge sufficient to form a belief as to the allegations of Paragraph 25 and accordingly deny the allegations of Paragraph 25.

26. Defendants admit the allegations of Paragraph 26.

27. Defendants admit the allegations of Paragraph 27.

28. Paragraph 28 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 28.

29. Paragraph 29 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 29.

30. Paragraph 30 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 30.

31. To the extent the complaint references specific allegations in separate actions, the pleadings of any such actions speak for themselves, and therefore no response is needed. Except as expressly admitted, Defendants deny the allegations of this paragraph.

32. Defendants admit that they directed patent licensing and enforcement activities towards Google and Cloudflare. Defendants otherwise are without information or knowledge

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

1 sufficient to form a belief as to the allegations of Paragraph 32 and accordingly deny the allegations
2 of Paragraph 32.

3 33. Defendants admit that they sent licensing communications to Google, which speak
4 for themselves. Defendants otherwise deny the allegations of Paragraph 33 and/or are without
5 information or knowledge sufficient to form a belief as to the allegations of Paragraph 33 and
6 accordingly deny the allegations of Paragraph 33.
7

8 34. Defendants are without information regarding who Google considers to be a
9 “customer,” and Google has not identified any specific customers or communications, and on that
10 basis cannot admit allegations relating to Google’s customers. Defendants otherwise deny the
11 allegations of Paragraph 34.

12 35. Defendants admit that they contracted with and acquired the Patents-in-Suit from
13 HPE. Defendants otherwise deny the allegations of Paragraph 35 and/or are without information or
14 knowledge sufficient to form a belief as to the allegations of Paragraph 35 and accordingly deny the
15 allegations of Paragraph 35.
16

17 36. Defendants admit that they engaged non-party PPS in connection with intellectual
18 property matters. Defendants admit that non-party PPS has an employee residing in this District.
19 Defendants otherwise deny the allegations of Paragraph 36.
20

21 37. Defendants admit that Valtrus is a wholly-owned subsidiary of Key Patent.
22 Defendants further admit that Key Patent has a financial interest in the licensing of Valtrus’ patent
23 portfolio and authorizes Valtrus’ licensing activities. Defendants otherwise deny the allegations of
24 Paragraph 37.

25 38. Defendants do not contest that venue is proper.

26 39. Paragraph 39 states a legal conclusion to which no response is required. To the extent
27 a response is required, Defendants deny the allegations of Paragraph 39.
28

DEFENDANTS’ ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF’S COMPLAINT

1 40. Defendants do not contest that this action is subject to assignment on a district-wide
2 basis.

3 **FIRST COUNT**

4 **(Declaratory Judgment of Non-Infringement of U.S. Patent No. 8,379,538)**

5 41. Defendants restate and incorporate by reference the responses in Defendants'
6 Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

7 42. Defendants admit the allegations of Paragraph 42.

8 43. Paragraph 43 states a legal conclusion to which no response is required. To the extent
9 a response is required, Defendants deny the allegations of Paragraph 43.

10 44. Paragraph 44 states a legal conclusion to which no response is required. To the extent
11 a response is required, Defendants deny the allegations of Paragraph 44.

12 45. Paragraph 45 states a legal conclusion to which no response is required. To the extent
13 a response is required, Defendants do not contest that a controversy currently exists regarding the
14 '538 Patent.

15 46. Defendants admit that Google seeks the relief stated in Paragraph 46, but otherwise
16 deny the allegations of Paragraph 46.

17 **SECOND COUNT**

18 **(Declaratory Judgment of Non-Infringement of U.S. Patent No. 7,251,588)**

19 47. Defendants restate and incorporate by reference the responses in Defendants'
20 Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

21 48. Defendants admit the allegations of Paragraph 48.

22 49. Paragraph 49 states a legal conclusion to which no response is required. To the extent
23 a response is required, Defendants deny the allegations of Paragraph 49.

24 DEFENDANTS' ANSWER, AFFIRMATIVE
25 DEFENSES, AND COUNTERCLAIMS TO
26 PLAINTIFF'S COMPLAINT

50. Paragraph 50 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 50.

51. Paragraph 51 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 51.

52. Defendants admit that Google seeks the relief stated in Paragraph 52, but otherwise deny the allegations of Paragraph 52.

THIRD COUNT

(Declaratory Judgment of Non-Infringement of U.S. Patent No. 8,370,416)

53. Defendants restate and incorporate by reference the responses in Defendants' Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

54. Defendants admit the allegations of Paragraph 54.

55. Paragraph 55 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 55.

56. Paragraph 56 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 56.

57. Paragraph 57 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 57.

58. Defendants admit that Google seeks the relief stated in Paragraph 58, but otherwise deny the allegations of Paragraph 58.

FOURTH COUNT

(Declaratory Judgment of Non-Infringement of U.S. Patent No. 7,640,332)

59. Defendants restate and incorporate by reference the responses in Defendants' Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

60. Defendants admit the allegations of Paragraph 60.

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

61. Paragraph 61 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 61.

62. Paragraph 62 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 62.

63. Paragraph 63 states a legal conclusion to which no response is required. To the extent a response is required, Defendants do not contest that a controversy currently exists regarding the '332 Patent.

64. Defendants admit that Google seeks the relief stated in Paragraph 64, but otherwise deny the allegations of Paragraph 64.

FIFTH COUNT

(Declaratory Judgment of Non-Infringement of U.S. Patent No. 7,904,686)

65. Defendants restate and incorporate by reference the responses in Defendants' Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

66. Defendants admit the allegations of Paragraph 66.

67. Paragraph 67 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 67.

68. Paragraph 68 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 68.

69. Paragraph 69 states a legal conclusion to which no response is required. To the extent a response is required, Defendants do not contest that a controversy currently exists regarding the '686 Patent.

70. Defendants admit that Google seeks the relief stated in Paragraph 70, but otherwise deny the allegations of Paragraph 70.

SIXTH COUNT

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

(Declaratory Judgment of Non-Infringement of U.S. Patent No. 7,730,218)

71. Defendants restate and incorporate by reference the responses in Defendants' Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

72. Defendants admit the allegations of Paragraph 72.

73. Paragraph 73 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 73.

74. Paragraph 74 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 74.

75. Paragraph 75 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 75.

76. Defendants admit that Google seeks the relief stated in Paragraph 76, but otherwise deny the allegations of Paragraph 76.

SEVENTH COUNT

(Declaratory Judgment of Non-Infringement of U.S. Patent No. 7,936,738)

77. Defendants restate and incorporate by reference the responses in Defendants' Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

78. Defendants admit the allegations of Paragraph 78.

79. Paragraph 79 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 79.

80. Paragraph 80 states a legal conclusion to which no response is required. To the extent a response is required, Defendants deny the allegations of Paragraph 80.

81. Paragraph 81 states a legal conclusion to which no response is required. To the extent a response is required, Defendants do not contest that a controversy currently exists regarding the '738 Patent.

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

1 82. Defendants admit that Google seeks the relief stated in Paragraph 82, but otherwise
2 deny the allegations of Paragraph 82.

3 83. Defendants admit that Google seeks the relief stated in Paragraph 83, but otherwise
4 deny the allegations of Paragraph 83.

5
6 **EIGHTH COUNT**

7 **(Declaratory Judgment of Non-Infringement of U.S. Patent No. 7,376,953)**

8 84. Defendants restate and incorporate by reference the responses in Defendants'
9 Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

10 85. Defendants admit the allegations of Paragraph 85.

11 86. Paragraph 86 states a legal conclusion to which no response is required. To the extent
12 a response is required, Defendants deny the allegations of Paragraph 86.

13 87. Paragraph 87 states a legal conclusion to which no response is required. To the extent
14 a response is required, Defendants deny the allegations of Paragraph 87.

15 88. Paragraph 88 states a legal conclusion to which no response is required. To the extent
16 a response is required, Defendants deny the allegations of Paragraph 88.

17 89. Defendants admit that Google seeks the relief stated in Paragraph 89, but otherwise
18 deny the allegations of Paragraph 89.

19
20 **NINTH COUNT**

21 **(Declaratory Judgment of Non-Infringement of U.S. Patent No. 6,889,244)**

22 90. Defendants restate and incorporate by reference the responses in Defendants'
23 Answer and the allegations in Defendants' Counterclaims as set forth fully herein.

24 91. Defendants admit the allegations of Paragraph 91.

25 92. Paragraph 92 states a legal conclusion to which no response is required. To the extent
26 a response is required, Defendants deny the allegations of Paragraph 92.
27
28

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

1 93. Paragraph 93 states a legal conclusion to which no response is required. To the extent
2 a response is required, Defendants deny the allegations of Paragraph 93.

3 94. Paragraph 94 states a legal conclusion to which no response is required. To the extent
4 a response is required, Defendants deny the allegations of Paragraph 94.

5 95. Defendants admit that Google seeks the relief stated in Paragraph 95, but otherwise
6 deny the allegations of Paragraph 95.

7
8 **FIRST AFFIRMATIVE DEFENSE**

9 **(Infringement)**

10 Google's declaratory judgment claims should be denied because Google has infringed,
11 directly and/or indirectly (including induced or contributorily), the Patents-in-Suit either literally or
12 under the doctrine of equivalents.

13
14 **SECOND AFFIRMATIVE DEFENSE**

15 **(Lack of Subject Matter Jurisdiction)**

16 Google's declaratory judgment claims should be dismissed for lack of a case or controversy,
17 including because Defendants have provided Google a covenant not to sue on U.S. Patent Nos.
18 7,251,588; 8,370,416; 7,730,218; 7,376,953; and 6,889,244.

19 **THIRD AFFIRMATIVE DEFENSE**

20 **(Failure to State a Claim)**

21 Google's declaratory judgment claims should be dismissed for failure to state facts sufficient
22 to constitute a cause of action against Defendants.

23
24 **FOURTH AFFIRMATIVE DEFENSE**

25 **(Unclean Hands)**

26 Google's declaratory judgment claims are barred by the doctrine of unclean hands.

27 **FIFTH AFFIRMATIVE DEFENSE**

28
DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

1 **(Estoppel)**

2 Google is estopped from pursuing its declaratory judgment claims by reason of their own
3 actions and course of conduct.

4 **SIXTH AFFIRMATIVE DEFENSE**

5 **(Waiver)**

6 Google's declaratory judgment claims are barred to the extent that Google waived its rights,
7 if any, to pursue them by reason of its own actions and course of conduct.
8

9 **SEVENTH AFFIRMATIVE DEFENSE**

10 **(Laches)**

11 Google's declaratory judgment claims are barred by the doctrine of laches.
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

COUNTERCLAIMS

Defendants and Counter-Plaintiffs Valtrus Innovations Ltd. and Key Patent Innovations Ltd., by and through their undersigned counsel, hereby plead the following Counterclaims for Patent Infringement against Plaintiff Google LLC and allege as follows:

THE PARTIES AND JURISDICTION

1. Defendant Valtrus is the successor in interest to a substantial patent portfolio created by Hewlett Packard Enterprise and its predecessor companies, including Compaq, Verity, and Hewlett-Packard Development Company (collectively, "HPE"). Valtrus is an Irish entity duly organized and existing under the laws of the Republic of Ireland. The address of the registered office of Valtrus is: The Glasshouses GH2, 92 Georges Street Lower, Dun Laoghaire, Dublin A96 VR66, Ireland.

2. Valtrus is the assignee of U.S. Patent Nos. 8,379,538 ("the '538 Patent"), 7,640,332 ("the '332 Patent"), 7,904,686 ("the '686 Patent"), 7,936,738 ("the '738 Patent"), and 7,057,509 ("the '509 Patent") (collectively, the "Asserted Patents").

3. Defendant Key Patent is the beneficiary of a trust pursuant to which Valtrus owns, holds, and asserts the Asserted Patents. Key Patent is an Irish entity duly organized and existing under the laws of the Republic of Ireland. The address of the registered office of Key Patent is: The Glasshouses GH2, 92 Georges Street Lower, Dun Laoghaire, Dublin A96 VR66, Ireland.

4. The Asserted Patents were developed by inventors working for HPE. HPE and its predecessors have been developing innovative search, computer processing, and server technology for decades.

5. On information and belief, Defendant Google is a limited liability company duly organized and existing under the laws of the State of Delaware, having a regular and established

1 place of business in the Northern District of California, including at the “Googleplex” located at
 2 1600 Amphitheatre Parkway, Mountain View, California 94043 in Santa Clara County.

3 6. These Counterclaims arise under the patent laws of the United States, 35 U.S.C. § 1
 4 et seq. Accordingly, this Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and
 5 1338(a).

7 **FIRST COUNTERCLAIM**

8 **(Infringement of U.S. Patent No. 7,640,332)**

9 7. Defendants re-allege and incorporate by reference the allegations of the preceding
 10 paragraphs of these Counterclaims as if fully set forth herein.

11 8. The '332 Patent, entitled “System and method for hot deployment/redeployment in
 12 grid computing environment,” was duly and lawfully issued on December 29, 2009. A true and
 13 correct copy of the '332 Patent is attached to the Complaint as Exhibit 4. *See* Dkt. 1, Ex. 4.

14 9. Defendants own by assignment the entire right, title, and interest in and to the '332
 15 Patent, including the exclusive right to seek damages for past, current, and future infringement
 16 thereof.

17 10. Defendants are informed and believe, and on this basis allege, that Google, agents of
 18 Google, and/or third parties acting under Google’s direction and control, have committed and
 19 continue to commit acts of direct infringement of one or more claims of the '332 Patent literally
 20 and/or under the doctrine of equivalents, by making, using, selling, offering to sell, and/or importing
 21 into the United States, Google Kubernetes Engine and all Google internal services that use
 22 Kubernetes or Google Kubernetes Engine. Google controls Google Kubernetes Engine and all
 23 Google internal services that use Kubernetes or Google Kubernetes Engine as a whole and obtains
 24 benefits from them.
 25
 26
 27
 28

DEFENDANTS’ ANSWER, AFFIRMATIVE
 DEFENSES, AND COUNTERCLAIMS TO
 PLAINTIFF’S COMPLAINT

11. For example, Google Kubernetes Engine embodies every limitation of at least claim 1 of the '332 Patent, literally or under the doctrine of equivalents, as set forth below. The further descriptions below, which are based on publicly available information, are preliminary examples and are non-limiting.

“1. A method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes, comprising:”

12. Google Kubernetes Engine uses a method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes.

13. For example, Google Kubernetes Engine uses a method of rolling updates to allow for hot deployment and redeployment (rollouts and rollbacks). Using rolling updates, application bundles, packaged in Pods, are updated and scheduled on grid nodes (e.g., Worker nodes). Worker nodes are virtual or physical machines, the collection of which are contained in the grid computing environment, known as a Cluster.

Automated rollouts and rollbacks

Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.

¹ <https://kubernetes.io/>

Core Components

A Kubernetes cluster consists of a control plane and one or more worker nodes. Here's a brief overview of the main components:

Control Plane Components

Manage the overall state of the cluster:

kube-apiserver

The core component server that exposes the Kubernetes HTTP API

etcd

Consistent and highly-available key value store for all API server data

kube-scheduler

Looks for Pods not yet bound to a node, and assigns each Pod to a suitable node.

kube-controller-manager

Runs controllers to implement Kubernetes API behavior.

cloud-controller-manager (optional)

Integrates with underlying cloud provider(s).

2

Node Components

Run on every node, maintaining running pods and providing the Kubernetes runtime environment:

kubelet

Ensures that Pods are running, including their containers.

kube-proxy (optional)

Maintains network rules on nodes to implement Services.

Container runtime

Software responsible for running containers. Read [Container Runtimes](#) to learn more.

3

Updating an application

Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones.

4

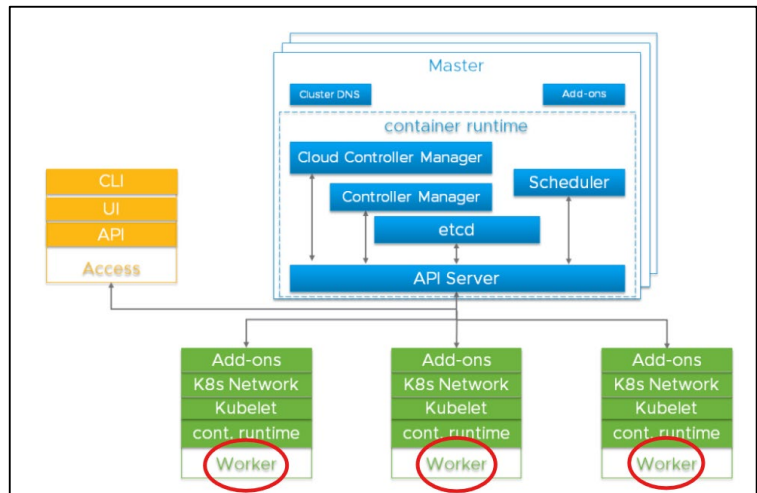
² <https://kubernetes.io/docs/concepts/overview/components/>

³ *Id.*

⁴ <https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/#updating-an-application>

A Pod always runs on a **Node**. A Node is a worker machine in Kubernetes and may be either a virtual or a physical machine, depending on the cluster. Each Node is managed by ⁵

A GKE cluster consists of a *control plane* and worker machines called *nodes*. The control plane and nodes make up the Kubernetes cluster orchestration system. GKE Autopilot manages the entire underlying ⁶



“adding a new version of an application release bundle in a repository server;”

14. Google Kubernetes Engine operates using a method further comprising adding a new version of an application release bundle in a repository server.

15. For example, users add a new version of an application release bundle by uploading a new container image, which are software bundles, in a repository server located within the container registry.

⁵ <https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/>

⁶ <https://docs.cloud.google.com/kubernetes-engine/docs/concepts/cluster-architecture>

⁷ <https://kubernetes.io/blog/2018/08/03/out-of-the-clouds-onto-the-ground-how-to-make-kubernetes-production-grade-anywhere/#critical-components-in-a-kubernetes-cluster>

Once you have a [running Kubernetes cluster](#), you can deploy your containerized applications on top of it. To do so, you create a Kubernetes **Deployment**. The [Deployment instructs Kubernetes how to create and update instances of your application](#). Once you've created a Deployment, the Kubernetes control plane schedules the application instances included in that Deployment to run on individual Nodes in the cluster.

When you create a [Deployment](#), you'll need to specify the [container image for your application](#) and the number of replicas that you want to run. You can change that information later by updating your

Images

A [container image](#) represents binary data that [encapsulates an application and all its software dependencies](#). Container images are [executable software bundles](#) that can run standalone and that make very well-defined assumptions about their runtime environment.

You typically [create a container image of your application and push it to a registry](#) before referring to it in a Pod.

As well as providing binary images, a [container registry](#) can also serve a [container image index](#). An image index can point to

If you don't specify a [registry hostname](#), Kubernetes assumes that you mean the [Docker public registry](#). You can change this behavior

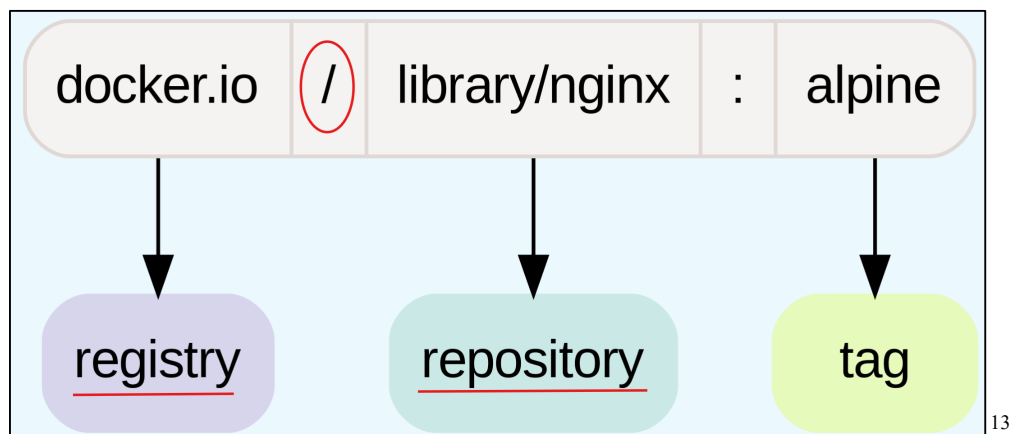
⁸ <https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/#kubernetes-deployments>

⁹ *Id.*

¹⁰ <https://kubernetes.io/docs/concepts/containers/images/>

¹¹ <https://kubernetes.io/docs/concepts/containers/images/#multi-architecture-images-with-image-indexes>

¹² <https://kubernetes.io/docs/concepts/containers/images/#image-names>



your.private.registry.example.com / janedoe/jdoe-private:v1

“determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server;”

16. Google Kubernetes Engine operates using a method further comprising determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server.

17. For example, upon a container image being updated in a Deployment, the Deployment Controller creates new ReplicaSets that determine which of the one or more grid nodes (e.g., Worker nodes) are running an application (e.g., a Pod containing the application) associated

¹³ <https://oras.land/docs/concepts/reference/>

¹⁴ <https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/#create-a-pod-that-uses-your-secret>

1 with the added new version. The new ReplicaSets version's ".spec.selector" labels match the
 2 previous version's, but the ".spec.template." fields vary, therefore indicating which Pods contain
 3 the previous version of the application. Since a Pod's lifetime is spent on a single Worker node, the
 4 ReplicaSet therefore determines the node(s) on which the application is running.

6 Deployments

7 A Deployment manages a set of Pods to run an
 8 application workload, usually one that doesn't maintain
 9 state.

10 A *Deployment* provides declarative updates for Pods and ReplicaSets.

11 You describe a *desired state* in a Deployment, and the Deployment
 12 Controller changes the actual state to the desired state at a
 13 controlled rate. You can define Deployments to create new
 14 ReplicaSets, or to remove existing Deployments and adopt all their
 15 resources with new Deployments.

15

16 A ReplicaSet is linked to its Pods via the Pods'
 17 metadata.ownerReferences field, which specifies what resource the
 18 current object is owned by. All Pods acquired by a ReplicaSet have their
 19 owning ReplicaSet's identifying information within their
 20 ownerReferences field. It's through this link that the ReplicaSet knows of
 21 the state of the Pods it is maintaining and plans accordingly.

16

26 ¹⁵ <https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/>

27 ¹⁶ [https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/#how-a-replicaset-](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/#how-a-replicaset-works)
 28 works

Rollover (aka multiple updates in-flight)

Each time a new Deployment is observed by the Deployment controller, a ReplicaSet is created to bring up the desired Pods. If the Deployment is updated, the existing ReplicaSet that controls Pods whose labels match `.spec.selector` but whose template does not match `.spec.template` is scaled down. Eventually, the new ReplicaSet is scaled to `.spec.replicas` and all old ReplicaSets is scaled to 0.

17

relatively ephemeral (rather than durable) entities. Pods are created, assigned a unique ID (UID), and scheduled to run on nodes where they remain until termination (according to restart policy) or deletion. If a

18

Kubernetes runs your workload by placing containers into Pods to run on *Nodes*. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the

19

“notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use;”

18. Google Kubernetes Engine operates using a method further comprising notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use.

19. For example, when a Deployment is updated with the new version of the application release bundle (e.g., an updated container image), the Deployment Controller creates a new ReplicaSet with an associated PodSpec. The client application manager—the Kubelet, which resides on a Worker node—is notified by the API Server of the new PodSpec, along with the data transfer protocol of the Open Container Initiative Distribution Specification.

¹⁷ <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#rollover-aka-multiple-updates-in-flight>

¹⁸ <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

¹⁹ <https://kubernetes.io/docs/concepts/architecture/nodes/>

Each time a new Deployment is observed by the Deployment controller, a ReplicaSet is created to bring up the desired Pods. If the Deployment is updated, the existing ReplicaSet that controls Pods whose labels match `.spec.selector` but whose template does not match `.spec.template` is scaled down. Eventually, the new ReplicaSet is scaled to `.spec.replicas` and all old ReplicaSets is scaled to 0.

20

A ReplicaSet is defined with fields, including a selector that specifies how to identify Pods it can acquire, a number of replicas indicating how many Pods it should be maintaining, and a pod template specifying the data of new Pods it should create to meet the number of replicas criteria. A ReplicaSet then fulfills its purpose by creating and deleting Pods as needed to reach the desired number. When a ReplicaSet needs to create new Pods, it uses its Pod template.

21

The kubelet is the primary "node agent" that runs on each node. It can register the node with the apiserver using one of: the hostname; a flag to override the hostname; or specific logic for a cloud provider.

The kubelet works in terms of a PodSpec. A PodSpec is a YAML or JSON object that describes a pod. The kubelet takes a set of PodSpecs that are provided through various mechanisms (primarily through the apiserver) and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

22

The Open Container Initiative Distribution Specification (a.k.a. "OCI Distribution Spec") defines an API protocol to facilitate and standardize the distribution of content.

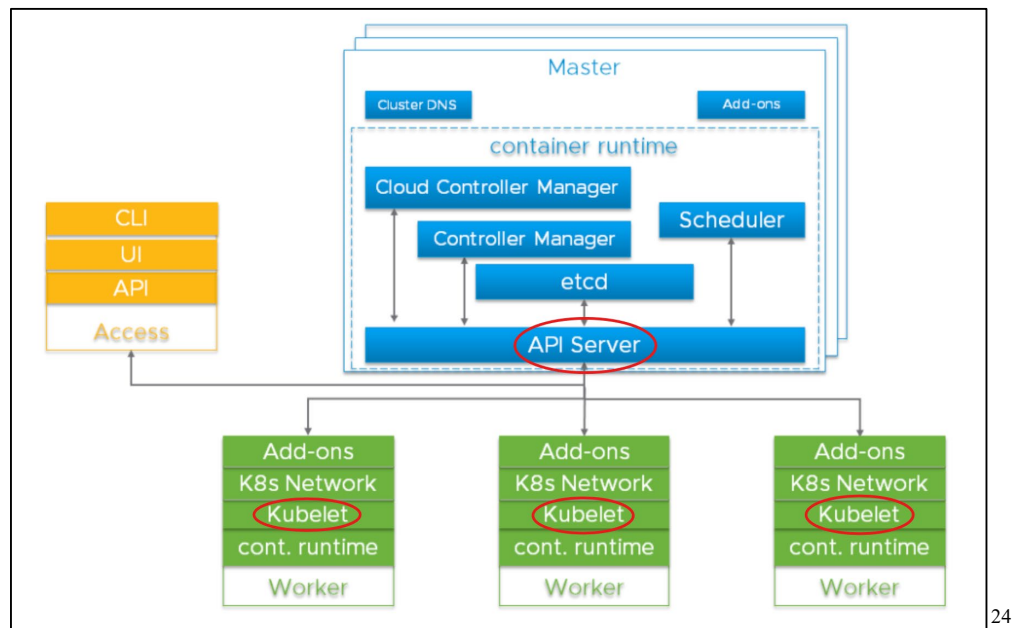
23

²⁰ <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#rollover-aka-multiple-updates-in-flight>

²¹ <https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/#how-a-replicaset-works>

²² <https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/#synopsis>

²³ <https://github.com/opencontainers/distribution-spec/blob/main/spec.md>



“hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.”

20. Google Kubernetes Engine operates using a method further comprising hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers.

21. For example, the client application manager—the Kubelet—reads the new PodSpec provided by the API server and instructs the hot deployment plug-in—the Container Runtime—to download and hot deploy/redeploy the appropriate image from the container registry onto the

²⁴ <https://kubernetes.io/blog/2018/08/03/out-of-the-clouds-onto-the-ground-how-to-make-kubernetes-production-grade-anywhere/#critical-components-in-a-kubernetes-cluster>

Worker nodes, which the Container Runtime does in accordance with the Open Container Initiative Distribution Specification data transfer protocol.

Updating an application

Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones.

25

Declare the new state of the Pods by updating the PodTemplateSpec of the Deployment. A new ReplicaSet is created, and the Deployment gradually scales it up while scaling down the old ReplicaSet, ensuring Pods are replaced at a controlled rate. Each new ReplicaSet updates the revision of the Deployment.

26

Rollover (aka multiple updates in-flight)

Each time a new Deployment is observed by the Deployment controller, a ReplicaSet is created to bring up the desired Pods. If the Deployment is updated, the existing ReplicaSet that controls Pods whose labels match `.spec.selector` but whose template does not match `.spec.template` is scaled down. Eventually, the new ReplicaSet is scaled to `.spec.replicas` and all old ReplicaSets is scaled to 0.

27

Step 2: Kubelet Communicates with the Container Runtime

- Kubelet retrieves the pod specification from the API Server.
- It instructs the container runtime (Docker, containerd, or CRI-O) to pull the nginx image and start the container.

28

²⁵ <https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/#updating-an-application>

²⁶ <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#use-case>

²⁷ <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#rollover-aka-multiple-updates-in-flight>

²⁸ <https://medium.com/careerbytecode/100days-of-k8s-with-devops-challenge-week-3recap-day-15-day-21-key-takeaways-d3747956643d>

Each container runtime has its own strengths, and many users have asked for Kubernetes to support more runtimes. In the Kubernetes 1.5 release, we are proud to introduce the Container Runtime Interface (CRI) -- a plugin interface which enables kubelet to use a wide variety of container runtimes, without the need to recompile. CRI consists of a protocol buffers and gRPC API, and

29

Initiative image. Any OCI-compliant image, regardless of the tool you use to build it, will look the same to Kubernetes. Both containerd and CRI-O know how to pull those images and run them. This is why we have a standard for

30

The Open Container Initiative Distribution Specification (a.k.a. "OCI Distribution Spec") defines an API protocol to facilitate and standardize the distribution of content.

31

The .spec.template field contains the following sub-fields:

- The Pods are labeled app: nginx using the .metadata.labels field.
- The Pod template's specification, or .spec field, indicates that the Pods run one container, nginx, which runs the nginx Docker Hub image at version 1.14.2.
- Create one container and name it nginx using the .spec.containers[0].name field.

32

²⁹ <https://kubernetes.io/blog/2016/12/container-runtime-interface-cri-in-kubernetes/>

³⁰ <https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/>

³¹ <https://github.com/opencontainers/distribution-spec/blob/main/spec.md>

³² <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#creating-a-deployment>

```

Name:          nginx-deployment
Namespace:     default
CreationTimestamp: Thu, 30 Nov 2017 10:56:25 +0000
Labels:       app=nginx
Annotations:  deployment.kubernetes.io/revision=2
Selector:     app=nginx
Replicas:     3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType: RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
    nginx:
      Image:      nginx:1.16.1
      Port:       80/TCP
      Environment: <none>
      Mounts:      <none>
      Volumes:     <none>
  Conditions:
    Type           Status    Reason
    ----           -
    Available       True      MinimumReplicasAvailable
    Progressing     True      NewReplicaSetAvailable
  OldReplicaSets: <none>
  NewReplicaSet:  nginx-deployment-1564180365 (3/3 replicas created)
  Events:
    Type      Reason      Age    From          Message
    ----      -
    Normal    ScalingReplicaSet    2m    deployment-controller    Scaled up replica set nginx-deployment-2035384211 to 3
    Normal    ScalingReplicaSet    24s   deployment-controller    Scaled up replica set nginx-deployment-1564180365 to 1
    Normal    ScalingReplicaSet    22s   deployment-controller    Scaled down replica set nginx-deployment-2035384211 to 2
    Normal    ScalingReplicaSet    22s   deployment-controller    Scaled up replica set nginx-deployment-1564180365 to 2
    Normal    ScalingReplicaSet    19s   deployment-controller    Scaled down replica set nginx-deployment-2035384211 to 1
    Normal    ScalingReplicaSet    19s   deployment-controller    Scaled up replica set nginx-deployment-1564180365 to 3
    Normal    ScalingReplicaSet    14s   deployment-controller    Scaled down replica set nginx-deployment-2035384211 to 0

```

33

Here you see that when you first created the Deployment, it created a ReplicaSet (nginx-deployment-2035384211) and scaled it up to 3 replicas directly. When you updated the Deployment, it created a new ReplicaSet (nginx-deployment-1564180365) and scaled it up to 1 and waited for it to come up. Then it scaled down the old ReplicaSet to 2 and scaled up the new ReplicaSet to 2 so that at least 3 Pods were available and at most 4 Pods were created at all times. It then continued scaling up and down the new and the old ReplicaSet, with the same rolling update strategy. Finally, you'll have 3 available replicas in the new ReplicaSet, and the old ReplicaSet is scaled down to 0.

34

³³ <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#updating-a-deployment>

³⁴ *Id.*

22. According to Google’s own documentation, Google “use[s] [Google Kubernetes Engine] for several” internal Google services, including Vertex AI and DeepMind.^{35,36} On information and belief, those services also include Google Ads and Google Ad Manager.³⁷

23. Google is inducing infringement of the ’332 Patent, in violation of 35 U.S.C. § 271(b), including by knowingly encouraging or aiding third parties to make, use, sell, or offer to sell Google Kubernetes Engine and all Google internal services that use Kubernetes or Google Kubernetes Engine in the United States, or to import Google Kubernetes Engine and all Google internal services that use Kubernetes or Google Kubernetes Engine into the United States, and/or in violation of 35 U.S.C. § 271(f)(1), by supplying or causing to be supplied Google Kubernetes Engine and all Google internal services that use Kubernetes or Google Kubernetes Engine in or from the United States, without license or authority from Defendants, with knowledge of or willful blindness to the fact that Google’s actions will induce others to directly infringe the ’332 Patent. Google actively encourages, directs, and controls third parties, including its end users, to make and/or use Google Kubernetes Engine and all Google internal services that use Kubernetes or Google Kubernetes Engine in an infringing manner, including through Google’s website, advertisements, documentation/instructional material, customer support, application and feature descriptions, and in-application instructions. Google has intentionally aided and encouraged third parties, including its end users, to use Google Kubernetes Engine and all Google internal services that use Kubernetes or Google Kubernetes Engine in the United States in a manner that it knows would infringe or have

³⁵ <https://cloud.google.com/blog/products/containers-kubernetes/why-choose-gke-as-your-kubernetes-service>

³⁶ <https://blog.google/innovation-and-ai/infrastructure-and-cloud/google-cloud/live-from-google-cloud-next-advancing-every-business-in-the-cloud/>

³⁷ <https://medium.com/@YodgorbekKomilo/unveiling-the-system-design-of-google-ads-architecture-scalability-and-reliability-a8f4adeb947d>.

1 a high probability of infringing the '332 Patent, with the specific intent that those performing the
2 acts infringe the '332 Patent.

3 24. Google is contributorily infringing the '332 Patent in violation of 35 U.S.C. § 271(c),
4 by selling, offering to sell, and/or importing into the United States, and/or in violation of 35 U.S.C.
5 § 271(f)(2), by supplying or causing to be supplied in or from the United States, Google Kubernetes
6 Engine and all Google internal services that use Kubernetes or Google Kubernetes Engine for use
7 in infringing the '332 Patent, constituting a material part of the invention, knowing Google
8 Kubernetes Engine and all Google internal services that use Kubernetes or Google Kubernetes
9 Engine to be especially made or especially adapted for use in infringing the '332 Patent. Google
10 Kubernetes Engine and all Google internal services that use Kubernetes or Google Kubernetes
11 Engine are not staple articles or commodities of commerce suitable for substantial non-infringing
12 use.
13

14
15 25. As a result of Google's infringement of the '332 Patent, Defendants have been
16 damaged. In addition, Google's infringing acts and practices have caused and are causing immediate
17 and irreparable harm to Defendants. Defendants are entitled to recover for damages sustained as a
18 result of Google's wrongful acts in an amount yet to be determined and to receive such other and
19 further relief, including equitable relief, as this Court deems just and proper.

20
21 26. Defendants are further informed, and on this basis allege, that Google's infringement
22 of the '332 Patent has been and continues to be deliberate and willful, and, therefore, this is an
23 exceptional case warranting an award of enhanced damages for up to three times the actual damages
24 awarded and attorney's fees to Defendants pursuant to 35 U.S.C. §§ 284-285. Google has had
25 knowledge of the '332 Patent and its infringement thereof, including through its knowledge of
26 Defendants' public assertions that others, including Home Depot, infringe the '332 Patent by using
27 Google Kubernetes Engine, *Valtrus Innovations Ltd. v. The Home Depot*, No. 2:25-cv-00081-JRG
28

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

(E.D. Tex. Jan. 27, 2025), and yet Google has deliberately continued to infringe in a wanton, malicious, and egregious manner, with reckless disregard for Defendants' patent rights.

SECOND COUNTERCLAIM

(Infringement of U.S. Patent No. 7,057,509)

27. Defendants re-allege and incorporate by reference the allegations of the preceding paragraphs of these Counterclaims as if fully set forth herein.

28. The '509 Patent, entitled "Monitoring an object with identification data and tracking data," was duly and lawfully issued on June 6, 2006. A true and correct copy of the '509 Patent is attached as Exhibit 1.

29. Defendants own by assignment the entire right, title, and interest in and to the '509 Patent, including the exclusive right to seek damages for past, current, and future infringement thereof.

30. Defendants are informed and believe, and on this basis allege, that Google, agents of Google, and/or third parties acting under Google's direction and control, have committed and continue to commit acts of direct infringement of one or more claims of the '509 Patent literally and/or under the doctrine of equivalents, by making, using, selling, offering to sell, and/or importing into the United States, Google Maps and Google Pixel. Google controls Google Maps and Google Pixel as a whole and obtains benefits from them.

31. For example, Google Maps and Google Pixel embody every limitation of at least claim 17 of the '509 Patent, literally or under the doctrine of equivalents, as set forth below. The further descriptions below, which are based on publicly available information, are preliminary examples and are non-limiting.

"17. A method of monitoring an object, said method comprising:"

32. Google Maps and Google Pixel use a method of monitoring an object.

33. For example, Google Maps' aggregating location analytics system uses a method of monitoring, tracking, and storing device data through Google Maps Timeline, and utilizing said data to provide insights to Google users. Google Pixel uses this method through Google Maps.

Manage your Google Maps Timeline

Timeline helps you go back in time and remember where you've been by automatically saving your visits and routes to your Google Maps Timeline on each of your signed-in devices.

To make Google experiences more useful for everyone, Timeline data can be used to

- show community trends, such as popular times and transit crowdedness, based on aggregated location data

Timeline data can also help nearby businesses estimate how often ads drive visits to their stores, if you have Web & App Activity turned on. Only anonymous estimates, not personal

“generating identification data associated with each monitored object located at an identification gateway of a monitored environment, wherein said generating identification data is performed by an identification system;”

34. Google Maps operates using a method further comprising generating identification data associated with each monitored object located at an identification gateway of a monitored environment, wherein said generating identification data is performed by an identification system.

35. For example, the Google Maps location tracking functionality generates a personal identifier based on user account information for each device being tracked at an identification

38

<https://support.google.com/maps/answer/6258979?hl=en&co=GENIE.Platform%3DDesktop>

39 <https://policies.google.com/technologies/location-data?hl=en>

40 *Id.*

1 gateway, such as a cellular base station or a Wi-Fi access point, that is within the Google Maps
2 monitored environment.

3 36. As a further example, Google Pixel uses the above-described method through its use
4 of Google Maps.

5
6 Timeline is turned off by default. If you turn on Timeline, your precise device location is
7 regularly saved, for every eligible mobile device where you're signed in. These device 41

8 **["generating tracking data associated with each monitored object located within said**
9 **monitored environment, wherein said generating tracking data is performed by a**
10 **tracking system;"]**

11 37. Google Maps operates using a method further comprising generating tracking data
12 associated with each monitored object located within said monitored environment, wherein said
13 generating tracking data is performed by a tracking system.

14 38. For example, the Google Maps location tracking functionality generates, on a regular
15 basis, tracking data (e.g., location data) of a monitored device within the Google Maps monitored
16 environment.

17 39. As a further example, Google Pixel uses the above-described method through its use
18 of Google Maps.

19
20 Your device location may also periodically be used in the
21 background. When Timeline is on, even when Google apps
22 aren't in use, your device's precise location is regularly
23 saved to your device. 42

24
25
26
27 ⁴¹ *Id.*

28 ⁴² <https://support.google.com/accounts/answer/14200149?hl=en-419>

“merging and storing said identification data and said tracking data of each monitored object to form monitoring data for each monitored object;”]

40. Google Maps operates using a method further comprising merging and storing said identification data and said tracking data of each monitored object to form monitoring data for each monitored object.

41. For example, on information and belief, Google Maps’ aggregating location analytics system merges the identification and tracking data (e.g., aggregates the personal identifier and location data) for each monitored device, which is then stored by Google Maps in the memory of the device on which it is running.

42. As a further example, Google Pixel uses the above-described method through its use of Google Maps.

Timeline

If you turn on Timeline, it will create a personal map on each of your devices that helps you remember places you’ve been, and routes and trips you’ve taken.

Timeline is turned off by default. If you turn on Timeline, your precise device location is regularly saved, for every eligible mobile device where you’re signed in. These device locations are used to build your Timeline, including when Google apps aren’t being used.

To make Google experiences more useful for everyone, Timeline data can be used to

- show community trends, such as popular times and transit crowdedness, based on aggregated location data
- detect and prevent fraud and abuse
- improve and provide Google services, including ads products

43

⁴³ <https://policies.google.com/technologies/location-data?hl=en>

Manage your Timeline data

Timeline  is a Google Account setting that helps you go back in time, and remember where you've been, by automatically saving your visits and routes to a map on each of your signed-in devices.

44

When you're signed in with your Google Account, each of your eligible devices will save visits and routes on its own Timeline. You also have the option to store an encrypted

45

“comparing new content of said monitoring data of a monitored object with prior content of said monitoring data of said monitored object.”]

43. Google Maps operates using a method further comprising comparing new content of said monitoring data of a monitored object with prior content of said monitoring data of said monitored object.

44. For example, Google Maps' aggregating location analytics system compares the aggregated data for a tracked device to determine when it arrived at a location, when it departed, and how often it visits said location. As a further example, the aggregating location analytics system compares the aggregated data for a tracked device to determine if an advertisement delivered to a device later drove the device user to physically visit the respective business.

45. As a further example, Google Pixel uses the above-described method through its use of Google Maps.

⁴⁴ <https://support.google.com/accounts/answer/14200149?hl=en-419>

⁴⁵ *Id.*

To make Google experiences more useful for everyone, Timeline data can be used to

- show community trends, such as popular times and transit crowdedness, based on aggregated location data

46

Timeline data can also help nearby businesses estimate how often ads drive visits to their stores, if you have Web & App Activity turned on. Only anonymous estimates, not personal

47

46. Google is inducing infringement of the '509 Patent, in violation of 35 U.S.C. § 271(b), including by knowingly encouraging or aiding third parties to make, use, sell, or offer to sell Google Maps and Google Pixel in the United States, or to import Google Maps and Google Pixel into the United States, and/or in violation of 35 U.S.C. § 271(f)(1), by supplying or causing to be supplied Google Maps and Google Pixel in or from the United States, without license or authority from Defendants, with knowledge of or willful blindness to the fact that Google's actions will induce others to directly infringe the '509 Patent. Google actively encourages, directs, and controls third parties, including its end users, to make and/or use Google Maps and Google Pixel in an infringing manner, including through Google's website, advertisements, documentation/instructional material, customer support, application and feature descriptions, and in-application instructions. Google has intentionally aided and encouraged third parties, including its end users, to use Google Maps and Google Pixel in the United States in a manner that it knows would infringe or have a high probability of infringing the '509 Patent, with the specific intent that those performing the acts infringe the '509 Patent.

47. Google is contributorily infringing the '509 Patent in violation of 35 U.S.C. § 271(c), by selling, offering to sell, and/or importing into the United States, and/or in violation of 35 U.S.C.

⁴⁶ <https://policies.google.com/technologies/location-data?hl=en>

⁴⁷ *Id.*

1 § 271(f)(2), by supplying or causing to be supplied in or from the United States, Google Maps and
 2 Google Pixel for use in infringing the '509 Patent, constituting a material part of the invention,
 3 knowing Google Maps and Google Pixel to be especially made or especially adapted for use in
 4 infringing the '509 Patent. Google Maps and Google Pixel are not staple articles or commodities of
 5 commerce suitable for substantial non-infringing use.

6
 7 48. As a result of Google's infringement of the '509 Patent, Defendants have been
 8 damaged. In addition, Google's infringing acts and practices have caused and are causing immediate
 9 and irreparable harm to Defendants. Defendants are entitled to recover for damages sustained as a
 10 result of Google's wrongful acts in an amount yet to be determined and to receive such other and
 11 further relief, including equitable relief, as this Court deems just and proper.

12 49. Defendants are further informed, and on this basis allege, that Google's infringement
 13 of the '509 Patent has been and continues to be deliberate and willful, and, therefore, this is an
 14 exceptional case warranting an award of enhanced damages for up to three times the actual damages
 15 awarded and attorney's fees to Defendants pursuant to 35 U.S.C. §§ 284-285. As noted above,
 16 Google has had knowledge of the '509 Patent and its infringement thereof, and yet has deliberately
 17 continued to infringe in a wanton, malicious, and egregious manner, with reckless disregard for
 18 Defendants' patent rights.

20 **THIRD COUNTERCLAIM**

21 **(Infringement of U.S. Patent No. 8,379,538)**

22
 23 50. Defendants re-allege and incorporate by reference the allegations of the preceding
 24 paragraphs of these Counterclaims as if fully set forth herein.

25 51. The '538 Patent, entitled "Model-driven monitoring architecture," was duly and
 26 lawfully issued on February 19, 2013. A true and correct copy of the '538 Patent is attached to the
 27 Complaint as Exhibit 1. *See* Dkt. 1, Ex. 1.
 28

DEFENDANTS' ANSWER, AFFIRMATIVE
 DEFENSES, AND COUNTERCLAIMS TO
 PLAINTIFF'S COMPLAINT

52. Defendants own by assignment the entire right, title, and interest in and to the '538 Patent, including the exclusive right to seek damages for past, current, and future infringement thereof.

53. Defendants are informed and believe, and on this basis allege, that Google, agents of Google, and/or third parties acting under Google's direction and control, have committed and continue to commit acts of direct infringement of one or more claims of the '538 Patent literally and/or under the doctrine of equivalents, by making, using, selling, offering to sell, and/or importing into the United States, Google Kubernetes Engine, all Google internal services that use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel. Google controls Google Kubernetes Engine, all Google internal services that use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel as a whole and obtains benefits from them.

54. For example, Google Kubernetes Engine embodies every limitation of at least claim 7 of the '538 Patent, literally or under the doctrine of equivalents, as set forth below. The further descriptions below, which are based on publicly available information, are preliminary examples and are non-limiting.

["7. A method comprising:"]

55. Google Kubernetes Engine uses a method. For example, Google Kubernetes Engine uses a method for Horizontal Pod Autoscaling to scale workload resources to match demand.

The Horizontal Pod Autoscaler changes the shape of your Kubernetes workload by automatically increasing or decreasing the number of Pods in response to the workload's CPU or memory consumption, or in response to custom metrics reported from within Kubernetes or external metrics from sources outside of your cluster.

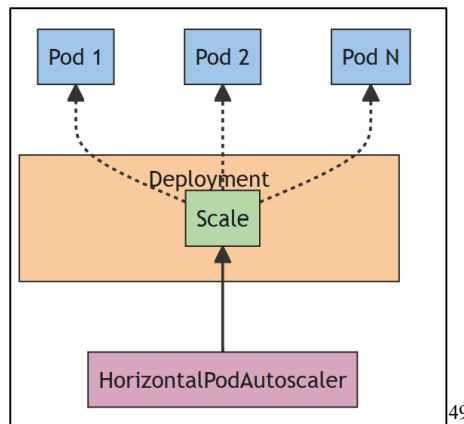
48

["providing a machine-readable monitoring model that defines configuration of a monitoring environment;"]

⁴⁸ <https://docs.cloud.google.com/kubernetes-engine/docs/concepts/horizontalpodautoscaler>

56. Google Kubernetes Engine operates using a method further comprising providing a machine-readable monitoring model that defines configuration of a monitoring environment.

57. For example, a machine-readable monitoring model—the HorizontalPodAutoscaler (“HPA”) functionality—defines the configuration of the Kubernetes autoscaling monitoring environment through a metric model containing parameters for memory and CPU usage, or other custom metrics.



Support for resource metrics

Any HPA target can be scaled based on the resource usage of the pods in the scaling target. When defining the pod specification the resource requests like cpu and memory should be specified. This is used to determine the resource utilization and used by the HPA controller to scale the target up or down. To use resource utilization based scaling specify a metric source like this:

⁴⁹ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#how-does-a-horizontalpodautoscaler-work>

⁵⁰ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#support-for-resource-metrics>

Scaling on custom metrics

① **FEATURE STATE:** Kubernetes v1.23 [stable]

(the `autoscaling/v2beta2` API version previously provided this ability as a beta feature)

Provided that you use the `autoscaling/v2` API version, you can configure a HorizontalPodAutoscaler to scale based on a custom metric (that is not built in to Kubernetes or any Kubernetes component). The HorizontalPodAutoscaler controller then queries for these custom metrics from the Kubernetes API.

51

“responsive to a change occurring in said monitoring environment’s configuration, said machine-readable monitoring model updating to reflect said change;”

58. Google Kubernetes Engine operates using a method further comprising, responsive to a change occurring in said monitoring environment’s configuration, said machine-readable monitoring model updating to reflect said change.

59. For example, responsive to an increase or decrease in the ratio between the desired and current CPU or memory usage (or other custom metric), the HPA Controller updates the Deployment’s scale subresource.

Once during each period, the controller manager queries the resource utilization against the metrics specified in each HorizontalPodAutoscaler definition. The controller manager

52

For per-pod resource metrics (like CPU), the controller fetches the metrics from the resource metrics API for each Pod targeted by the HorizontalPodAutoscaler. Then, if a target utilization value is set, the controller calculates the utilization value as a percentage of the equivalent resource request on the containers in each Pod. If a target raw value is set, the raw metric values are used directly. The controller then

53

⁵¹ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#scaling-on-custom-metrics>

⁵² <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#how-does-a-horizontalpodautoscaler-work>

⁵³ *Id.*

From the most basic perspective, the HorizontalPodAutoscaler controller operates on the ratio between desired metric value and current metric value:

$$desiredReplicas = \text{ceil} \left[currentReplicas \times \frac{currentMetricValue}{desiredMetricValue} \right]$$

54

Tolerance

① **FEATURE STATE:** Kubernetes v1.35 [beta](enabled by default)

The tolerance field configures a threshold for metric variations, preventing the autoscaler from scaling for changes below that value.

This tolerance is defined as the amount of variation around the desired metric value under which no scaling will occur. For example, consider a HorizontalPodAutoscaler configured with a target memory consumption of 100MiB and a scale-up tolerance of 5%:

55

The HorizontalPodAutoscaler controller accesses corresponding workload resources that support scaling (such as Deployments and StatefulSet). These resources each have a subresource named scale, an interface that allows you to dynamically set the number of replicas and examine each of their current states. For general information about subresources in the Kubernetes API, see [Kubernetes API Concepts](#).

56

Autoscaling during rolling update

Kubernetes lets you perform a rolling update on a Deployment. In that case, the Deployment manages the underlying ReplicaSets for you. When you configure autoscaling for a Deployment, you bind a HorizontalPodAutoscaler to a single Deployment. The HorizontalPodAutoscaler manages the replicas field of the Deployment. The deployment controller is responsible for setting the replicas of the underlying ReplicaSets so that they add up to a suitable number during the rollout and also afterwards.

57

⁵⁴ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#algorithm-details>

⁵⁵ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#tolerance>

⁵⁶ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#how-does-a-horizontalpodautoscaler-work>

⁵⁷ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#autoscaling-during-rolling-update>

“an element of said monitoring environment reading said machine-readable model and autonomously adapting its operation to the changed configuration.”]

60. Google Kubernetes Engine operates using a method further comprising an element of said monitoring environment reading said machine-readable model and autonomously adapting its operation to the changed configuration.

61. For example, the Deployment Controller, which is an element of the Kubernetes autoscaling environment, reads the HPA model’s update to the Deployment’s scale subresource and autonomously adapts its operation to add or remove replicas of the underlying ReplicaSets so as to increase or decrease the number of Pods deployed.

Autoscaling during rolling update

Kubernetes lets you perform a rolling update on a Deployment. In that case, the Deployment manages the underlying ReplicaSets for you. When you configure autoscaling for a Deployment, you bind a HorizontalPodAutoscaler to a single Deployment. The HorizontalPodAutoscaler manages the `replicas` field of the Deployment. The deployment controller is responsible for setting the `replicas` of the underlying ReplicaSets so that they add up to a suitable number during the rollout and also afterwards.

If you perform a rolling update of a StatefulSet that has an autoscaled number of replicas, the StatefulSet directly manages its set of Pods (there is no intermediate resource similar to ReplicaSet).

58

Roughly speaking, the HPA controller will increase and decrease the number of replicas (by updating the Deployment) to maintain an average CPU utilization across all Pods of 50%. The Deployment then updates the ReplicaSet - this is part of how all Deployments work in Kubernetes - and then the ReplicaSet either adds or removes Pods based on the change to its `.spec`.

59

⁵⁸ <https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/#autoscaling-during-rolling-update>

⁵⁹ <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/#create-horizontal-pod-autoscaler>

62. According to Google's own documentation, Google "use[s] [Google Kubernetes Engine] for several" internal Google services, including Vertex AI and DeepMind.^{60,61} On information and belief, those services also include Google Ads and Google Ad Manager.⁶²

63. In addition, Google Maps and Google Pixel embody every limitation of at least claim 7 of the '538 Patent, literally or under the doctrine of equivalents, as set forth below. The further descriptions below, which are based on publicly available information, are preliminary examples and are non-limiting.

"7. A method comprising:"

64. Google Maps and Google Pixel use a method. For example, Google Maps uses a method for sharing location on a smartphone, and Google Pixel uses a method for sharing location through Google Maps.

How to track someone on Google Maps

1. Start Google Maps on your iPhone or Android.
2. Tap your account avatar at the top right of the screen. In the pop-up menu, tap "Location sharing."
3. You'll see a list of everyone who is currently sharing their location with you at the bottom of the screen. To see someone's precise location, tap the entry and the map will zoom into their position.

63

⁶⁰ <https://cloud.google.com/blog/products/containers-kubernetes/why-choose-gke-as-your-kubernetes-service>

⁶¹ <https://blog.google/innovation-and-ai/infrastructure-and-cloud/google-cloud/live-from-google-cloud-next-advancing-every-business-in-the-cloud/>

⁶² <https://medium.com/@YodgorbekKomilo/unveiling-the-system-design-of-google-ads-architecture-scalability-and-reliability-a8f4adeb947d>.

⁶³ <https://www.businessinsider.com/reference/how-to-track-someone-on-google-maps>



64

["providing a machine-readable monitoring model that defines configuration of a monitoring environment;"]

65. Google Maps and Google Pixel operate using a method further comprising providing a machine-readable monitoring model that defines configuration of a monitoring environment.

66. For example, Google Maps' Android or iOS/macOS device-readable location tracking functionality defines the metric model that includes location tracking data of the Google Maps environment.

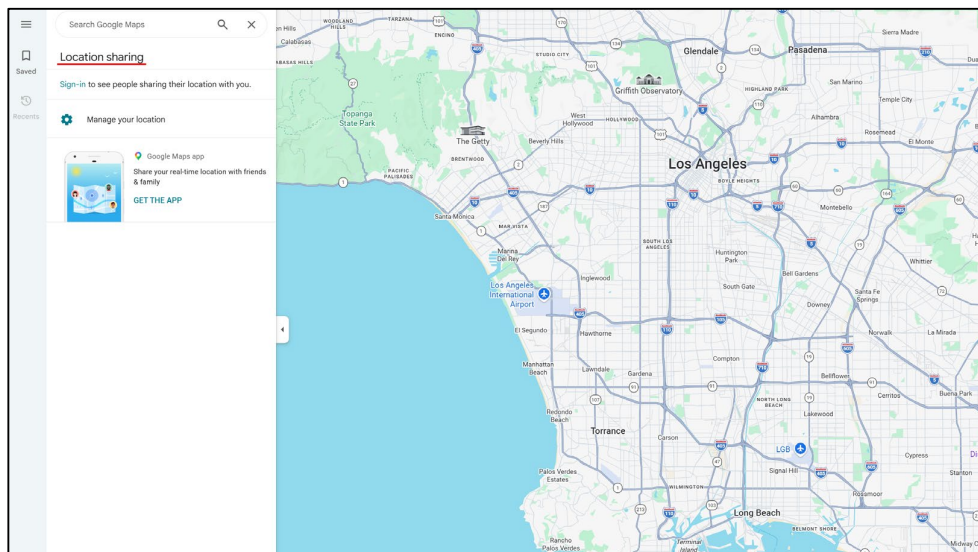
67. As a further example, Google Pixel uses the above-described method through its use of Google Maps.

⁶⁴ *Id.*

How to track someone on Google Maps

1. Start Google Maps on your iPhone or Android.
2. Tap your account avatar at the top right of the screen. In the pop-up menu, tap "Location sharing."
3. You'll see a list of everyone who is currently sharing their location with you at the bottom of the screen. To see someone's precise location, tap the entry and the map will zoom into their position.

65



66

“responsive to a change occurring in said monitoring environment’s configuration, said machine-readable monitoring model updating to reflect said change:”

68. Google Maps and Google Pixel operate using a method further comprising, responsive to a change occurring in said monitoring environment’s configuration, said machine-readable monitoring model updating to reflect said change.

⁶⁵ *Id.*

⁶⁶ https://www.google.com/maps/@34.0310106,-118.3860425,11z?authuser=0&entry=tu&g_ep=EgoyMDI2MDMwOS4wIKXMDSoASAFQAw%3D%3D

69. For example, responsive to a change in the location tracking data in the Google Maps environment, such as inclusion or exclusion of the location data on a person turning on or off location sharing, Google Maps' location tracking functionality updates the list of tracked people for location data gathering and display.

70. As a further example, Google Pixel uses the above-described method through its use of Google Maps.

Stop sharing your location with a specific person

1. Open your device's Settings app.
2. Tap **Location** > **Location services** > **Google Location Sharing**.
3. Tap the list of people you share your location with.
4. Find the person you want to stop sharing with. Next to the person's name, tap Stop.

67

Pause Location Sharing

You can pause any updates to your location on your device as long as the setting is turned off.

1. Open your device's Settings app.
2. Tap **Location** > **Location services** > **Google Location Sharing**.
3. Turn off Use Location Sharing.

Tips:

- This won't stop any of your shares, but they won't be updated with your real-time location until you turn this setting on.
- You can use other location services on your device, such as Google Maps, while this setting is turned off.
- If anyone else has an account on the device, it won't affect any shares they have. Each account on a device can manage their location shares separately.
- If you haven't shared your location with any other users, you can't turn this setting on or off. It shows as off by default.

68

“an element of said monitoring environment reading said machine-readable model and autonomously adapting its operation to the changed configuration.”

67

<https://support.google.com/accounts/answer/9363497?hl=en&co=GENIE.Platform%3DAndroid&oco=1>

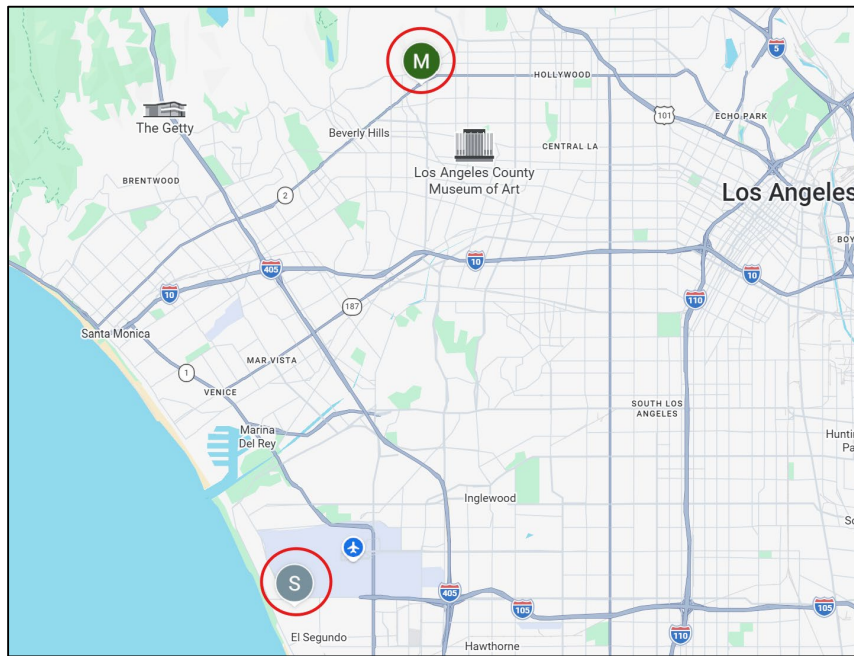
68 *Id.*

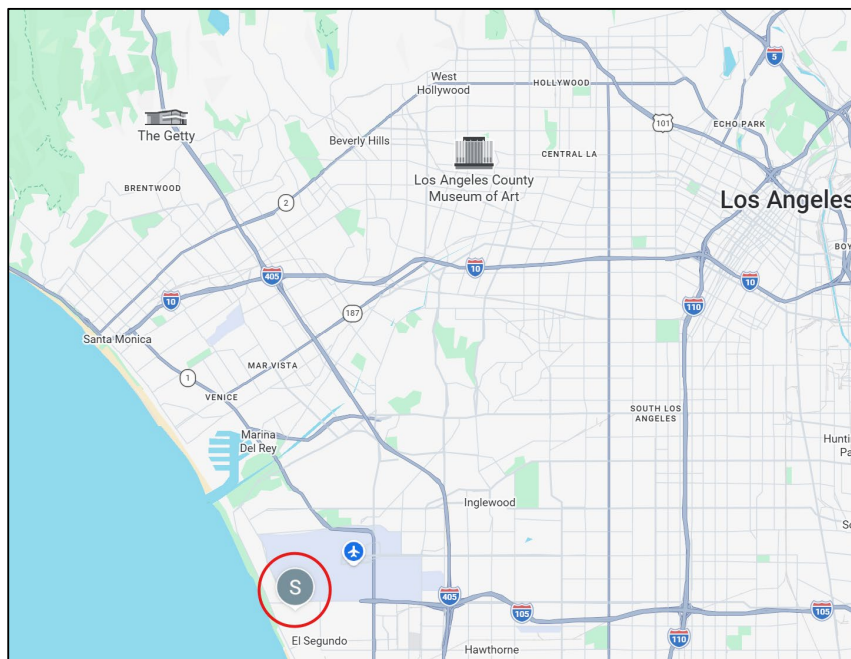
DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

1 71. Google Maps and Google Pixel operate using a method further comprising an
2 element of said monitoring environment reading said machine-readable model and autonomously
3 adapting its operation to the changed configuration.

4 72. For example, an element of the Google Maps environment reads the Android or
5 iOS/macOS device-readable location tracking functionality and autonomously adapts to display
6 locations for only those who are currently sharing their locations.

7 73. As a further example, Google Pixel uses the above-described method through its use
8 of Google Maps.





74. Google is inducing infringement of the '538 Patent, in violation of 35 U.S.C. § 271(b), including by knowingly encouraging or aiding third parties to make, use, sell, or offer to sell Google Kubernetes Engine, all Google internal services that use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel in the United States, or to import Google Kubernetes Engine, all Google internal services that use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel into the United States, and/or in violation of 35 U.S.C. § 271(f)(1), by supplying or causing to be supplied Google Kubernetes Engine, all Google internal services that use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel in or from the United States, without license or authority from Defendants, with knowledge of or willful blindness to the fact that Google's actions will induce others to directly infringe the '538 Patent. Google actively encourages, directs, and controls third parties, including its end users, to make and/or use Google Kubernetes Engine, all Google internal services that use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel in an infringing manner, including through Google's website, advertisements, documentation/instructional material, customer support, application and feature

1 descriptions, and in-application instructions. Google has intentionally aided and encouraged third
2 parties, including its end users, to use Google Kubernetes Engine, all Google internal services that
3 use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel in the United States
4 in a manner that it knows would infringe or have a high probability of infringing the '538 Patent,
5 with the specific intent that those performing the acts infringe the '538 Patent.
6

7 75. Google is contributorily infringing the '538 Patent in violation of 35 U.S.C. § 271(c),
8 by selling, offering to sell, and/or importing into the United States, and/or in violation of 35 U.S.C.
9 § 271(f)(2), by supplying or causing to be supplied in or from the United States, Google Kubernetes
10 Engine, all Google internal services that use Kubernetes or Google Kubernetes Engine, Google
11 Maps, and Google Pixel for use in infringing the '538 Patent, constituting a material part of the
12 invention, knowing Google Kubernetes Engine, all Google internal services that use Kubernetes or
13 Google Kubernetes Engine, Google Maps, and Google Pixel to be especially made or especially
14 adapted for use in infringing the '538 Patent. Google Kubernetes Engine, all Google internal
15 services that use Kubernetes or Google Kubernetes Engine, Google Maps, and Google Pixel are not
16 staple articles or commodities of commerce suitable for substantial non-infringing use.
17

18 76. As a result of Google's infringement of the '538 Patent, Defendants have been
19 damaged. In addition, Google's infringing acts and practices have caused and are causing immediate
20 and irreparable harm to Defendants. Defendants are entitled to recover for damages sustained as a
21 result of Google's wrongful acts in an amount yet to be determined and to receive such other and
22 further relief, including equitable relief, as this Court deems just and proper.
23

24 77. Defendants are further informed, and on this basis allege, that Google's infringement
25 of the '538 Patent has been and continues to be deliberate and willful, and, therefore, this is an
26 exceptional case warranting an award of enhanced damages for up to three times the actual damages
27 awarded and attorney's fees to Defendants pursuant to 35 U.S.C. §§ 284-285. On May 13, 2021,
28

1 Defendants provided Google with a letter informing Google of the '538 Patent and its infringement
 2 thereof, and yet Google has deliberately continued to infringe in a wanton, malicious, and egregious
 3 manner, with reckless disregard for Defendants' patent rights.

4 **FOURTH COUNTERCLAIM**

5 **(Infringement of U.S. Patent No. 7,904,686)**

6
 7 78. Defendants re-allege and incorporate by reference the allegations of the preceding
 8 paragraphs of these Counterclaims as if fully set forth herein.

9 79. The '686 Patent, entitled "Data Security for Use With a File System," was duly and
 10 lawfully issued on March 8, 2011. A true and correct copy of the '686 Patent is attached to the
 11 Complaint as Exhibit 5. *See* Dkt. 1, Ex. 5.

12 80. Defendants own by assignment the entire right, title, and interest in and to the '686
 13 Patent, including the exclusive right to seek damages for past, current, and future infringement
 14 thereof.

15
 16 81. Defendants are informed and believe, and on this basis allege, that Google, agents of
 17 Google, and/or third parties acting under Google's direction and control, have committed and
 18 continue to commit acts of direct infringement of one or more claims of the '686 Patent literally
 19 and/or under the doctrine of equivalents, by making, using, selling, offering to sell, and/or importing
 20 into the United States, Google's Managed Apache Hadoop with Google Dataproc ("Google
 21 Dataproc"). Google controls Google Dataproc as a whole and obtains benefits from it.

22
 23 82. For example, Google Dataproc embodies every limitation of at least claim 1 of the
 24 '686 Patent, literally or under the doctrine of equivalents, as set forth below. The further descriptions
 25 below, which are based on publicly available information, are preliminary examples and are non-
 26 limiting.

“1. A method of providing data security for use with a file system, the method comprising:”]

83. Google Dataproc uses a method of providing data security for use with a file system. For example, Google Dataproc integrates and uses the Apache Hadoop Distributed File System (HDFS) for storage:

Home > Documentation > Data analytics > Dataproc > Guides

Was this helpful?

Dataproc Hadoop data storage

[Send feedback](#)

Dataproc integrates with Apache Hadoop and the Hadoop Distributed File System (HDFS). The following features and considerations can be important when selecting compute and data storage options for Dataproc clusters and jobs:

- HDFS with Cloud Storage: Dataproc uses the Hadoop Distributed File System (HDFS) for storage. Additionally, Dataproc automatically installs the HDFS-compatible [Cloud Storage connector](#), which enables the use of Cloud Storage in parallel with HDFS. Data can be moved in and out of a cluster through upload and download to HDFS or Cloud Storage.
- VM disks:
 - By default, when no local SSDs are provided, HDFS data and intermediate shuffle data is stored on VM boot disks, which are [Persistent Disks](#).
 - If you use [local SSDs](#), HDFS data and intermediate shuffle data is stored on the SSDs.
 - Persistent disk (PD) size and type affect performance and VM size, whether using HDFS or Cloud Storage for data storage.
 - **VM Boot disks are deleted when the cluster is deleted.**

69

84. Apache Hadoop framework provides security features to protect data present in Hadoop clusters. Hadoop uses the HDFS file system to store data. HDFS comprises a name node and multiple data nodes:

Introduction

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is part of the Apache Hadoop Core project. The project URL is <http://hadoop.apache.org/>

70

⁶⁹ <https://docs.cloud.google.com/dataproc/docs/concepts/dataproc-hdfs>

⁷⁰ <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>

NameNode and DataNodes

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

71

The Communication Protocols

All HDFS communication protocols are layered on top of the TCP/IP protocol. A client establishes a connection to a configurable TCP port on the NameNode machine. It talks the ClientProtocol with the NameNode. The DataNodes talk to the NameNode using the DataNode Protocol. A Remote Procedure Call (RPC) abstraction wraps both the Client Protocol and the DataNode Protocol. By design, the NameNode never initiates any RPCs. Instead, it only responds to RPC requests issued by DataNodes or clients.

72

“applying a mapping function to data block numbers that are associated with a file, wherein the data block numbers are contained in an index node associated with said file;”

85. Google Dataproc performs the step of applying a mapping function to data block numbers that are associated with a file wherein the data block numbers are contained in an index node associated with said file.

86. For example, NameNode performs various operation such as opening, closing of files and mapping data block locations (data block number). Mapping (mapping function) is the process of determining which data block is stored at which location:

NameNode and DataNodes

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

73

⁷¹ *Id.*

⁷² *Id.*

⁷³ *Id.*

The Persistence of File System Metadata

The HDFS namespace is stored by the NameNode. The NameNode uses a transaction log called the EditLog to persistently record every change that occurs to file system metadata. For example, creating a new file in HDFS causes the NameNode to insert a record into the EditLog indicating this. Similarly, changing the replication factor of a file causes a new record to be inserted into the EditLog. The NameNode uses a file in its local host OS file system to store the EditLog. The entire file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the FsImage. The FsImage is stored as a file in the NameNode's local file system too.

The NameNode keeps an image of the entire file system namespace and file Blockmap in memory. When the NameNode starts up, or a checkpoint is triggered by a configurable threshold, it reads the FsImage and EditLog from disk, applies all the transactions from the EditLog to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk. It can then truncate the old EditLog because its transactions have been applied to the persistent FsImage. This process is called a checkpoint. The purpose of a checkpoint is to make sure that HDFS has a consistent view of the file system metadata by taking a snapshot of the file system metadata and saving it to FsImage. Even though it is efficient to read a FsImage, it is not efficient to make incremental edits directly to a FsImage. Instead of modifying FsImage for each edit, we persist the edits in the EditLog. During the checkpoint the changes from EditLog are applied to the FsImage. A checkpoint can be triggered at a given time interval (`dfs.namenode.checkpoint.period`) expressed in seconds, or after a given number of filesystem transactions have accumulated (`dfs.namenode.checkpoint.txns`). If both of these properties are set, the first threshold to be reached triggers a checkpoint.

74

87. Data blocks are associated with a file. The data block locations are stored in the Name node's (index node) memory:

The DataNode stores HDFS data in files in its local file system. The DataNode has no knowledge about HDFS files. It stores each block of HDFS data in a separate file in its local file system. The DataNode does not create all files in the same directory. Instead, it uses a heuristic to determine the optimal number of files per directory and creates subdirectories appropriately. It is not optimal to create all local files in the same directory because the local file system might not be able to efficiently support a huge number of files in a single directory. When a DataNode starts up, it scans through its local file system, generates a list of all HDFS data blocks that correspond to each of these local files, and sends this report to the NameNode. The report is called the Blockreport.

75

"obtaining mapped data block numbers after applying the mapping function,**wherein the mapped data block numbers are addresses of data of the file in a storage device."]**

88. Google Dataproc performs the step of obtaining mapped data block numbers after applying the mapping function, wherein the mapped data block numbers are addresses of data of the file in a storage device.

89. For example, Name Node determines 'the addresses of the data nodes' (mapped data block numbers) on which the data block is stored. Data nodes (storage device) store the data blocks of a file:

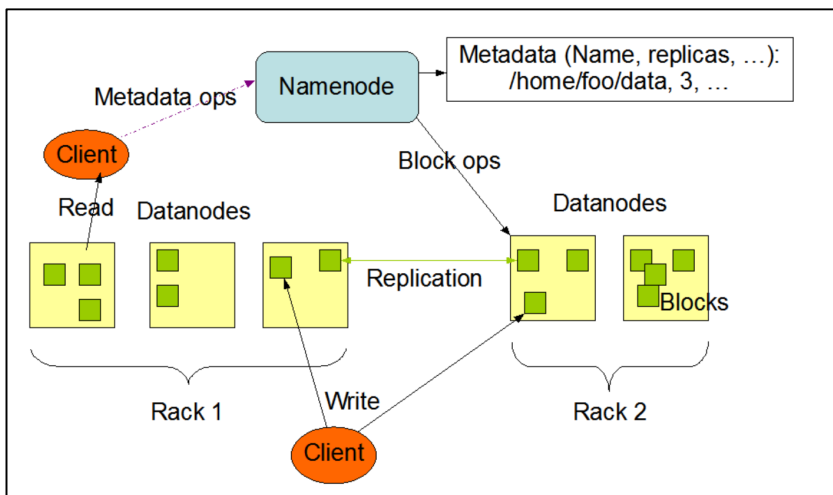
⁷⁴ *Id.*

⁷⁵ *Id.*

NameNode and DataNodes

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

76



77

90. Google is inducing infringement of the '686 Patent, in violation of 35 U.S.C. § 271(b), including by knowingly encouraging or aiding third parties to make, use, sell, or offer to sell Google Dataproc in the United States, or to import Google Dataproc into the United States, and/or in violation of 35 U.S.C. § 271(f)(1), by supplying or causing to be supplied Google Dataproc in or from the United States, without license or authority from Defendants, with knowledge of or willful blindness to the fact that Google's actions will induce others to directly infringe the '686 Patent. Google actively encourages, directs, and controls third parties, including its end users, to make and/or use Google Dataproc in an infringing manner, including through Google's website, advertisements, documentation/instructional material, customer support, application and feature descriptions, and in-application instructions. Google has intentionally aided and encouraged third parties, including its end users, to use Google Dataproc in the United States in a manner that it

⁷⁶ *Id.*

⁷⁷ *Id.*

1 knows would infringe or have a high probability of infringing the '686 Patent, with the specific
2 intent that those performing the acts infringe the '686 Patent.

3 91. Google is contributorily infringing the '686 Patent in violation of 35 U.S.C. § 271(c),
4 by selling, offering to sell, and/or importing into the United States, and/or in violation of 35 U.S.C.
5 § 271(f)(2), by supplying or causing to be supplied in or from the United States, Google Dataproc
6 for use in infringing the '686 Patent, constituting a material part of the invention, knowing Google
7 Dataproc to be especially made or especially adapted for use in infringing the '686 Patent. Google
8 Dataproc is not a staple article or commodity of commerce suitable for substantial non-infringing
9 use.
10

11 92. As a result of Google's infringement of the '686 Patent, Defendants have been
12 damaged. In addition, Google's infringing acts and practices have caused and are causing immediate
13 and irreparable harm to Defendants. Defendants are entitled to recover for damages sustained as a
14 result of Google's wrongful acts in an amount yet to be determined and to receive such other and
15 further relief, including equitable relief, as this Court deems just and proper.
16

17 93. Defendants are further informed, and on this basis allege, that Google's infringement
18 of the '686 Patent has been and continues to be deliberate and willful, and, therefore, this is an
19 exceptional case warranting an award of enhanced damages for up to three times the actual damages
20 awarded and attorney's fees to Defendants pursuant to 35 U.S.C. §§ 284-285. As noted above,
21 Google has had knowledge of the '686 Patent and its infringement thereof, including through its
22 awareness of Defendants' action against NetApp for its use of Apache Hadoop, *Valtrus Innovations*
23 *Ltd. v. NetApp, Inc.*, No. 2:25-cv-00517 (E.D. Tex. May 9, 2025), and yet Google has deliberately
24 continued to infringe in a wanton, malicious, and egregious manner, with reckless disregard for
25 Defendants' patent rights.
26

27 **FIFTH COUNTERCLAIM**
28

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

(Infringement of U.S. Patent No. 7,936,738)

94. Defendants re-allege and incorporate by reference the allegations of the preceding paragraphs of these Counterclaims as if fully set forth herein.

95. The '738 Patent, entitled "Fault Tolerant Systems," was duly and lawfully issued on May 3, 2011. The '738 Patent is entitled to a priority date of July 6, 2004. A true and correct copy of the '738 Patent is attached to the Complaint as Exhibit 7. *See* Dkt. 1, Ex. 7.

96. Defendants own by assignment the entire right, title, and interest in and to the '738 Patent, including the exclusive right to seek damages for past, current, and future infringement thereof.

97. Defendants are informed and believe, and on this basis allege, that Google, agents of Google, and/or third parties acting under Google's direction and control, have committed and continue to commit acts of direct infringement of one or more claims of the '738 Patent literally and/or under the doctrine of equivalents, by making, using, selling, offering to sell, and/or importing into the United States, Google's Managed Service for Apache Kafka ("Kafka Service") and similar services such as Google's Pub/Sub Messaging Service ("Pub/Sub") and other similarly operated services like Google Cloud Workflow. Google controls the Kafka Service as a whole and obtains benefits from it and similar services such as Pub/Sub, which on information and belief functions in a substantially similar manner to Kafka. Google also uses Pub/Sub for internal services such as Google Ads, Search, and Gmail.⁷⁸

⁷⁸ <https://docs.cloud.google.com/pubsub/architecture> ("Pub/Sub is an asynchronous messaging service designed to be highly reliable and scalable. The service is built on a core Google infrastructure component that many Google products have relied upon for over a decade. **Google products including Ads, Search and Gmail use this infrastructure to send over 500 million messages per second, totaling over 1TB/s of data.** This article describes the salient design features that enables Pub/Sub to provide this type of scale reliably.")

1 98. For example, Google's Kafka Service embodies every limitation of at least claim 1
2 of the '738 Patent, literally or under the doctrine of equivalents, as set forth below. The further
3 descriptions below, which are based on publicly available information, are preliminary examples
4 and are non-limiting.

5 **["1. A method of storing context information in an outgoing message sent from a**
6 **node including a computing device using a protocol stack having at least one**
7 **layer, comprising:"]**

8
9 99. Google's Managed Service for Apache Kafka uses a method of storing context
10 information in an outgoing message sent from a node including a computing device using a protocol
11 stack having at least one layer.

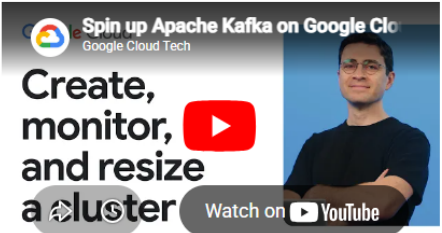
12 100. Google's Managed Service for Apache Kafka helps users run Apache Kafka clusters,
13 and runs the same software as Apache Kafka:
14

Home > Documentation > Data analytics > Google Cloud Managed Service for Apache Kafka > Guides Was this helpful?  

Managed Service for Apache Kafka overview

Send feedback

Managed Service for Apache Kafka is a Google Cloud service that helps you run secure, scalable open source Apache Kafka clusters. This page is an overview of what the service automates and simplifies for you. For more information about Apache Kafka, see the [Apache Kafka website](#).



Simple sizing and scaling

To size or scale a Managed Service for Apache Kafka cluster, you need only set the total vCPU count and RAM size for the cluster. Management of brokers, including storage, is fully automated. To keep up with demands of clients, you can monitor vCPU and other resource utilization and adjust them up or down.

When you set the vCPU count and RAM size, the service automates broker resizing and provisioning. If increasing the cluster size requires a new broker, the service can automatically rebalance partitions across brokers.

79

Compatible because we run Apache Kafka

Finally, Managed Service for Apache Kafka runs the same [open source software](#) you may already run in your environment. You don't have to change your application code to migrate it to the service.

80

101. Kafka stores context information in outgoing messages sent from a node including a computing device using a protocol stack having at least one layer. As a non-limiting example, Kafka stores context information such as topic name and partition index in outgoing Kafka messages:

⁷⁹ <https://docs.cloud.google.com/managed-service-for-apache-kafka/docs/overview>

⁸⁰ *Id.*

Main Concepts and Terminology

An **event** records the fact that “something happened” in the world or in your business. It is also called record or message in the documentation. When you read or write data to Kafka, you do this in the form of events. Conceptually, an event has a key, value, timestamp, and optional metadata headers. Here’s an example event:

- Event key: “Alice”
- Event value: “Made a payment of \$200 to Bob”
- Event timestamp: “Jun. 25, 2020 at 2:06 p.m.”

Producers are those client applications that publish (write) events to Kafka, and **consumers** are those that subscribe to (read and process) these events. In Kafka, producers and consumers are fully decoupled and agnostic of each other, which is a key design element to achieve the high scalability that Kafka is known for. For example, producers never need to wait for consumers. Kafka provides various [guarantees](#) such as the ability to process events exactly-once.

Events are organized and durably stored in **topics**. Very simplified, a topic is similar to a folder in a filesystem, and the events are the files in that folder. An example topic name could be “payments”. Topics in Kafka are always multi-producer and multi-subscriber: a topic can have zero, one, or many producers that write events to it, as well as zero, one, or many consumers that subscribe to these events. Events in a topic can be read as often as needed—unlike traditional messaging systems, events are not deleted after consumption. Instead, you define for how long Kafka should retain your events through a per-topic configuration setting, after which old events will be discarded. Kafka’s performance is effectively constant with respect to data size, so storing data for a long time is perfectly fine.

Topics are **partitioned**, meaning a topic is spread over a number of “buckets” located on different Kafka brokers. This distributed placement of your data is very important for scalability because it allows client applications to both read and write the data from/to many brokers at the same time. When a new event is published to a topic, it is actually appended to one of the topic’s partitions. Events with the same event key (e.g., a customer or vehicle ID) are written to the same partition, and Kafka [guarantees](#) that any consumer of a given topic-partition will always read that partition’s events in exactly the same order as they were written.

81

Produce API (Key: 0):

Requests:

```
Produce Request (Version: 3) => transactional_id acks timeout_ms [topic_data]
transactional_id => NULLABLE_STRING
acks => INT16
timeout_ms => INT32
topic_data => name [partition_data]
name => STRING
partition_data => index records
index => INT32
records => RECORDS
```

Request header version: 1

FIELD	DESCRIPTION
transactional_id	The transactional ID, or null if the producer is not transactional.
acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete. Allowed values: 0 for no acknowledgments, 1 for only the leader and -1 for the full ISR.
timeout_ms	The timeout to await a response in milliseconds.
<u>topic_data</u>	Each topic to produce to.
name	The topic name.
partition_data	Each partition to produce to.
<u>index</u>	The partition index.
records	The record data to be produced.

82

⁸¹ <https://kafka.apache.org/42/getting-started/introduction/>

⁸² <https://kafka.apache.org/42/design/protocol/#the-messages>

102. The outgoing Kafka messages are sent from a node including a computing device using a protocol stack having at least one layer (e.g., the stack used with Kafka, which may, for example, include adapter, transport (TCP), and messaging layers):

How does Kafka work in a nutshell?

Kafka is a distributed system consisting of **servers** and **clients** that communicate via a high-performance [TCP network protocol](#). It can be deployed on bare-metal hardware, virtual machines, and containers in on-premise as well as cloud environments.

Servers : Kafka is run as a cluster of one or more servers that can span multiple datacenters or cloud regions. Some of these servers form the storage layer, called the brokers. Other servers run [Kafka Connect](#) to continuously import and export data as event streams to integrate Kafka with your existing systems such as relational databases as well as other Kafka clusters. To let you implement mission-critical use cases, a Kafka cluster is highly scalable and fault-tolerant: if any of its servers fails, the other servers will take over their work to ensure continuous operations without any data loss.

Clients : They allow you to write distributed applications and microservices that read, write, and process streams of events in parallel, at scale, and in a fault-tolerant manner even in the case of network problems or machine failures. Kafka ships with some such clients included, which are augmented by [dozens of clients](#) provided by the Kafka community: clients are available for Java and Scala including the higher-level [Kafka Streams](#) library, for Go, Python, C/C++, and many other programming languages as well as REST APIs.

83

103. Google's Pub/Sub similarly uses a method of storing context information in an outgoing message sent from a node including a computing device using a protocol stack having at least one layer:

⁸³ <https://kafka.apache.org/protocol>; <https://kafka.apache.org/42/getting-started/introduction/>

Home > Documentation > Data analytics > Pub/Sub > Guides

Was this helpful?

What is Pub/Sub?

[Send feedback](#)

Key Point: Understand what is Pub/Sub, why do businesses require Pub/Sub, and the advantages of Pub/Sub compared to similar technologies. Also, learn about core Pub/Sub concepts that include the terms topic, publisher, and subscriber.

Pub/Sub is an asynchronous and scalable messaging service that decouples services producing messages from services processing those messages.

Pub/Sub allows services to communicate asynchronously, with latencies typically on the order of 100 milliseconds.

Pub/Sub is used for streaming analytics and data integration pipelines to load and distribute data. It's equally effective as a messaging-oriented middleware for service integration or as a queue to parallelize tasks.

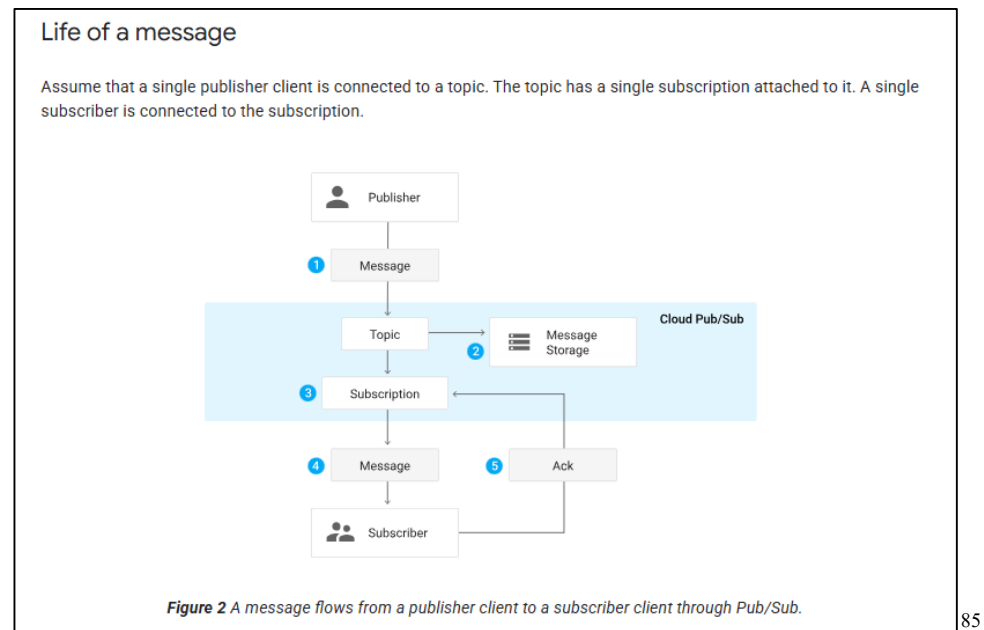
Pub/Sub lets you create systems of event producers and consumers, called **publishers** and **subscribers**. Publishers communicate with subscribers asynchronously by broadcasting events, rather than by synchronous remote procedure calls (RPCs).

Publishers send events to the Pub/Sub service, without regard to how or when these events are to be processed. Pub/Sub then delivers events to all the services that react to them. In systems communicating through RPCs, publishers must wait for subscribers to receive the data. However, the asynchronous integration in Pub/Sub increases the flexibility and robustness of the overall system.

To get started with Pub/Sub, check out the [Quickstart using Google Cloud console](#). For a more comprehensive introduction, see [Building a Pub/Sub messaging system](#).



84



⁸⁴ <https://docs.cloud.google.com/pubsub/docs/overview>

⁸⁵ <https://docs.cloud.google.com/pubsub/docs/publish-message-overview>

104. As a non-limiting example, Pub/Sub stores context information such as topic name and attributes in outgoing messages:

Use attributes to publish a message

You can embed custom attributes as metadata in Pub/Sub messages. Attributes are used to provide additional information about the message, such as its priority, origin, or destination. Attributes can also be used to filter messages on the subscription.

Follow these guidelines for using attributes in your messages:

- You can have at most 100 attributes per message.
- Attribute keys and values must be string types. There is no required encoding.
- Attribute keys must not start with `goog` and must not exceed 256 bytes.
- Attribute values must not exceed 1024 bytes.

The message schema can be represented as follows:

```
{
  "data": string,
  "attributes": {
    string: string,
    ...
  },
  "messageId": string,
  "publishTime": string,
  "orderingKey": string
}
```

For publish-side duplicates, it's possible to see different `publishTime` values for the same client-side original message, even with the same `messageId`.

The `PubsubMessage` JSON schema is published as part of the [REST](#) and [RPC](#) documentation. You can use custom attributes for event timestamps.

86

⁸⁶ <https://docs.cloud.google.com/pubsub/docs/publisher>

The following samples demonstrate how to publish a message with attributes to a topic.

Console gcloud C++ C# Go Java Node.js Python Ruby

Before trying this sample, follow the Python setup instructions in [Quickstart: Using Client Libraries](#). For more information, see the [Pub/Sub Python API reference documentation](#).

```
from google.cloud import pubsub_v1

# TODO(developer)
# project_id = "your-project-id"
# topic_id = "your-topic-id"

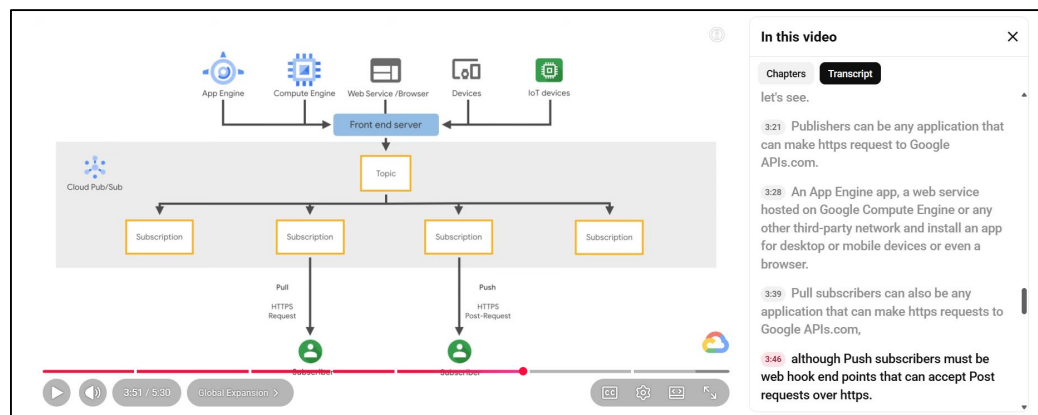
publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path(project_id, topic_id)

for n in range(1, 10):
    data_str = f"Message number {n}"
    # Data must be a bytestring
    data = data_str.encode("utf-8")
    # Add two attributes, origin and username, to the message
    future = publisher.publish(
        topic_path, data, origin="python-sample", username="gcp"
    )
    print(future.result())

print(f"Published messages with custom attributes to {topic_path}.")
```

87

105. The outgoing Pub/Sub messages are sent from a node including a computing device using a protocol stack having at least one layer (e.g., the stack used with Pub/Sub, which may, for example, include application, transport, and messaging layers):



88

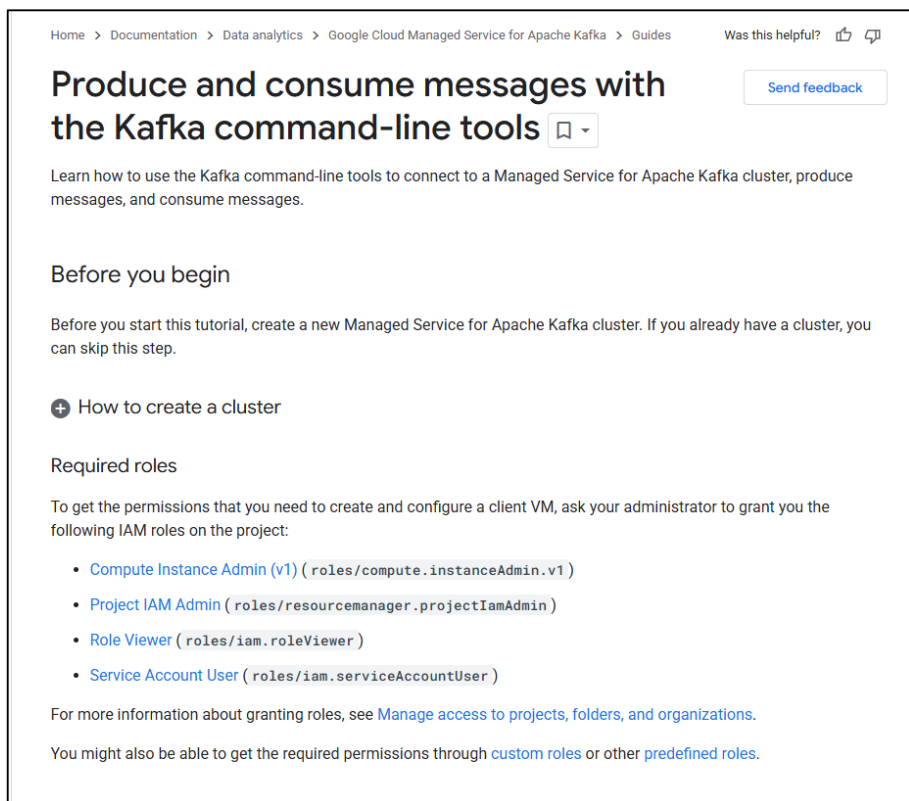
⁸⁷ *Id.*

⁸⁸ <https://www.youtube.com/watch?v=MjEam95VLiI&t=1s>

l“providing, by the computing device, the outgoing message from an application to a layer of the protocol stack, the outgoing message is destined for an application on a destination node;”]

106. Google’s Managed Service for Apache Kafka performs the step of providing, by the computing device, the outgoing message from an application to a layer of the protocol stack, the outgoing message is destined for an application on a destination node.

107. As a non-limiting example, Google’s Managed Kafka Service provides the outgoing message from an application (e.g., a client VM or Kafka Connect) to a layer of the protocol stack used with Kafka. As one example, Google explains below how to configure a client VM application and produce messages to a Managed Kafka Service cluster:



89

⁸⁹ <https://docs.cloud.google.com/managed-service-for-apache-kafka/docs/quickstart>

Produce and consume messages

From your SSH session, run the following commands to produce and consume Kafka messages.

1. Set the bootstrap address as an environment variable.

```
export BOOTSTRAP=bootstrap.CLUSTER_ID.REGION.managedkafka.PROJECT_ID.cloud.goog:9092
```

Replace the following:

- *CLUSTER_ID*: the name of your cluster
- *REGION*: where you created the cluster
- *PROJECT_ID*: your project ID

For more information, see [Get the bootstrap address](#).

2. List the topics in the cluster.

```
kafka-topics.sh --list \
  --bootstrap-server $BOOTSTRAP \
  --command-config client.properties
```

3. Write a message to a topic.

```
echo "hello world" | kafka-console-producer.sh \
  --topic KAFKA_TOPIC_NAME \
```

90

108. As another example, Google offers Kafka Connect as part of its Managed Kafka Service. Google's Kafka Service servers provide the outgoing message from Kafka Connect to a layer of the protocol stack used with Kafka:

⁹⁰ *Id.*

Home > Documentation > Data analytics > Google Cloud Managed Service for Apache Kafka > Guides Was this helpful?  

Kafka Connect overview

[Send feedback](#)

Kafka Connect is the preferred tool for data integration for Kafka developers. It provides a framework for connecting Kafka with external systems such as databases, message queues, and file systems.

Kafka Connect provides a curated set of built-in connector plugins, vetted and maintained by Google Cloud. These connector plugins are automatically patched and upgraded, simplifying maintenance and ensuring compatibility. Google Cloud also provides built-in monitoring and logging to maintain the health of your pipelines.

Kafka Connect APIs are offered as part of the Google Cloud Managed Service for Apache Kafka service. These APIs are accessible through managedkafka.googleapis.com and are integrated into the Google Cloud console and client libraries. To manage Kafka Connect, you can use the Google Cloud console, the gcloud CLI, the Managed Kafka API, the cloud client libraries, or Terraform.

Kafka Connect use cases

Kafka Connect supports data integration between your Managed Service for Apache Kafka cluster and various other systems. Here are some key use cases:

- Migrate your existing Kafka deployments to Managed Service for Apache Kafka.
- Replicate your Managed Service for Apache Kafka cluster to another region for disaster recovery.
- Stream data from Managed Service for Apache Kafka to BigQuery, Cloud Storage, Pub/Sub.

91

109. The Kafka stack includes at least adapter, messaging, and transport layers (TCP for transport):

Preliminaries

Network

Kafka uses a binary protocol over TCP. The protocol defines all APIs as request response message pairs. All messages are size delimited and are made up of the following primitive types.

The client initiates a socket connection and then writes a sequence of request messages and reads back the corresponding response message. No handshake is required on connection or disconnection. TCP is happier if you maintain persistent connections used for many requests to amortize the cost of the TCP handshake, but beyond this penalty connecting is pretty cheap.

The client will likely need to maintain a connection to multiple brokers, as data is partitioned and the clients will need to talk to the server that has their data. However it should not generally be necessary to maintain multiple connections to a single broker from a single client instance (i.e. connection pooling).

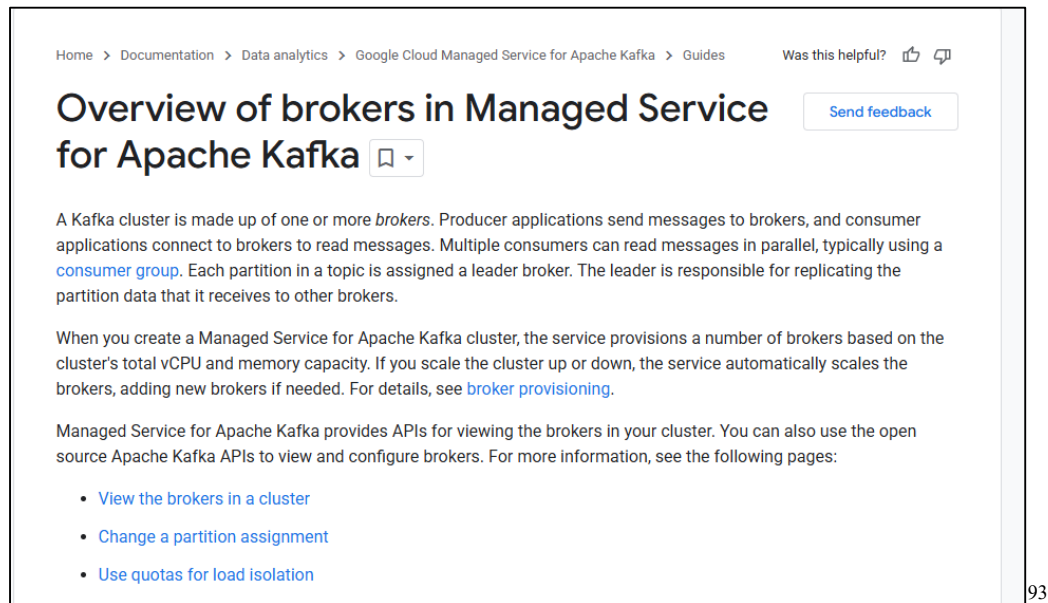
The server guarantees that on a single TCP connection, requests will be processed in the order they are sent and responses will return in that order as well. The broker's request processing allows only a single in-flight request per connection in order to guarantee this ordering. Note that clients can (and ideally should) use non-blocking IO to implement request pipelining and achieve higher throughput, i.e., clients can send requests even while awaiting responses for preceding requests since the outstanding requests will be buffered in the underlying OS socket buffer. All requests are initiated by the client, and result in a corresponding response message from the server except where noted.

92

⁹¹ <https://docs.cloud.google.com/managed-service-for-apache-kafka/docs/kafka-connect-overview>

⁹² <https://kafka.apache.org/protocol>

110. The outgoing message is destined for an application on a destination node, for example, an application running in the Kafka broker:



111. Google's Pub/Sub performs the step of providing, by the computing device, the outgoing message from an application to a layer of the protocol stack, the outgoing message is destined for an application on a destination node:

⁹³ <https://docs.cloud.google.com/managed-service-for-apache-kafka/docs/brokers>

Life of a message

Assume that a single publisher client is connected to a topic. The topic has a single subscription attached to it. A single subscriber is connected to the subscription.

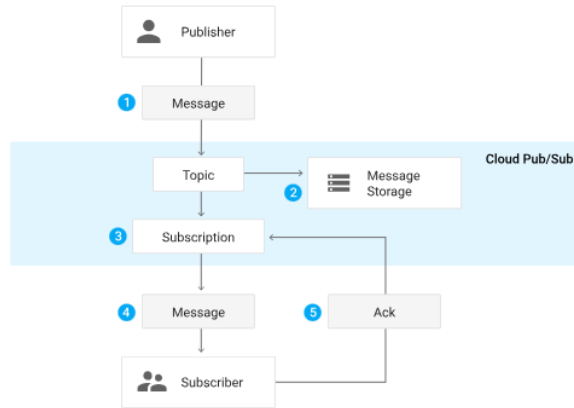


Figure 2 A message flows from a publisher client to a subscriber client through Pub/Sub.

94

“selectively indicating to the layer of the protocol stack that context information is to be obtained for that layer;”

112. Google’s Kafka Service performs the step of selectively indicating to the layer of the protocol stack that context information is to be obtained for that layer.

113. For example, Google’s Kafka Service allows configuration of message headers including context information such as the topic. Kafka will then append the context information to the outgoing message:

⁹⁴ <https://docs.cloud.google.com/pubsub/docs/publish-message-overview>

1

2

3

4

5

6

7

8

9

10

11

12

13

Produce and consume messages

From your SSH session, run the following commands to produce and consume Kafka messages.

1. Set the bootstrap address as an environment variable.

```
export BOOTSTRAP=bootstrap.CLUSTER_ID.REGION.managedkafka.PROJECT_ID.cloud.goog:9092
```

Replace the following:

- *CLUSTER_ID*: the name of your cluster
- *REGION*: where you created the cluster
- *PROJECT_ID*: your project ID

For more information, see [Get the bootstrap address](#).

2. List the topics in the cluster.

```
kafka-topics.sh --list \
--bootstrap-server $BOOTSTRAP \
--command-config client.properties
```

3. Write a message to a topic.

```
echo "hello world" | kafka-console-producer.sh \
--topic KAFKA_TOPIC_NAME \
```

95

14 114. Additionally, Kafka uses a particular request function to select and obtain different
15 context information (e.g., partition index) depending on the particular request, as shown below:

16

17

18

19

20

21

22

23

24

25

26

27

28

Produce API (Key: 0):

Requests:

```
Produce Request (Version: 3) => transactional_id acks timeout_ms [topic_data]
transactional_id => NULLABLE_STRING
acks => INT16
timeout_ms => INT32
topic_data => name [partition_data]
name => STRING
partition_data => index records
index => INT32
records => RECORDS
```

Request header version: 1

FIELD	DESCRIPTION
transactional_id	The transactional ID, or null if the producer is not transactional.
acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete. Allowed values: 0 for no acknowledgments, 1 for only the leader and -1 for the full ISR.
timeout_ms	The timeout to await a response in milliseconds.
topic_data	Each topic to produce to.
name	The topic name.
partition_data	Each partition to produce to.
index	The partition index.
records	The record data to be produced.

96

⁹⁵ <https://docs.cloud.google.com/managed-service-for-apache-kafka/docs/quickstart>

⁹⁶ <https://kafka.apache.org/42/design/protocol/#the-messages>

Requests:

```
Produce Request (version: 3) => transactional_id acks timeout_ms [topic_data]
transactional_id => NULLABLE_STRING
acks => INT16
timeout_ms => INT32
topic_data => name [partition_data]
name => STRING
partition_data => index records
index => INT32
records => RECORDS
```

Request header version: 1

FIELD	DESCRIPTION
transactional_id	The transactional ID, or null if the producer is not transactional.
acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete. Allowed values: 0 for no acknowledgments, 1 for only the leader and -1 for the full ISR.
timeout_ms	The timeout to await a response in milliseconds.
topic_data	Each topic to produce to.
name	The topic name.
partition_data	Each partition to produce to.
index	The partition index.
records	The record data to be produced.

```
Produce Request (version: 4) => transactional_id acks timeout_ms [topic_data]
transactional_id => NULLABLE_STRING
acks => INT16
timeout_ms => INT32
topic_data => name [partition_data]
name => STRING
partition_data => index records
index => INT32
records => RECORDS
```

97

115. As another example, Kafka Connect determines that a message should be directed to a Kafka Service Cluster and modified for use with the Kafka protocol, and thereby selectively indicates to the layer of the stack that context information is to be obtained for that layer.

116. For example, Kafka Connect allows configuration and modification of message headers including context information such as the topic:

Included transformations

Several widely-applicable data and routing transformations are included with Kafka Connect:

- Cast - Cast fields or the entire key or value to a specific type
- DropHeaders - Remove headers by name
- ExtractField - Extract a specific field from Struct and Map and include only this field in results
- Filter - Removes messages from all further processing. This is used with a predicate to selectively filter certain messages
- Flatten - Flatten a nested data structure
- HeaderFrom - Copy or move fields in the key or value to the record headers
- HoistField - Wrap the entire event as a single field inside a Struct or a Map
- InsertField - Add a field using either static data or record metadata
- InsertHeader - Add a header using static data
- MaskField - Replace field with valid null value for the type (0, empty string, etc) or custom replacement (non-empty string or numeric value only)
- RegexRouter - modify the topic of a record based on original topic, replacement string and a regular expression
- ReplaceField - Filter or rename fields
- SetSchemaMetadata - modify the schema name or version
- TimestampConverter - Convert timestamps between different formats
- TimestampRouter - Modify the topic of a record based on original topic and timestamp. Useful when using a sink that needs to write to different tables or indexes based on timestamps
- ValueToKey - Replace the record key with a new key formed from a subset of fields in the record value

98

⁹⁷ <https://kafka.apache.org/42/design/protocol/#the-messages>

⁹⁸ <https://kafka.apache.org/42/kafka-connect/user-guide/>

117. Kafka Connect also allows modification of the content of message headers (including adding, editing, and deleting headers):

Included transformations

Several widely-applicable data and routing transformations are included with Kafka Connect:

- Cast - Cast fields or the entire key or value to a specific type
- DropHeaders - Remove headers by name
- ExtractField - Extract a specific field from Struct and Map and include only this field in results
- Filter - Removes messages from all further processing. This is used with a predicate to selectively filter certain messages
- Flatten - Flatten a nested data structure
- HeaderFrom - Copy or move fields in the key or value to the record headers
- HoistField - Wrap the entire event as a single field inside a Struct or a Map
- InsertField - Add a field using either static data or record metadata
- InsertHeader - Add a header using static data
- MaskField - Replace field with valid null value for the type (0, empty string, etc) or custom replacement (non-empty string or numeric value only)
- RegexRouter - modify the topic of a record based on original topic, replacement string and a regular expression
- ReplaceField - Filter or rename fields
- SetSchemaMetadata - modify the schema name or version
- TimestampConverter - Convert timestamps between different formats
- TimestampRouter - Modify the topic of a record based on original topic and timestamp. Useful when using a sink that needs to write to different tables or indexes based on timestamps
- ValueToKey - Replace the record key with a new key formed from a subset of fields in the record value

99

118. Google Pub/Sub also performs the step of selectively indicating to the layer of the protocol stack that context information is to be obtained for that layer. As one non-limiting example, Google Pub/Sub does this through the use of SMTs, which are optional. If SMTs are enabled for a topic, Pub/Sub selectively indicates to a layer of the stack that context information is to be obtained for that layer:

⁹⁹ *Id.*

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

Home > Documentation > Data analytics > Pub/Sub > Guides

Was this helpful?  

Single Message Transforms (SMTs) overview

[Send feedback](#)

Most streaming data pipelines require data transformations. Some users prefer transforming data after it reaches its destination in an extract, load, transform (ELT) pipeline, while others opt for transforming data before its ingestion in an extract, transform, and load (ETL) pipeline. Traditionally, this architecture required complex pipelines with tools like Dataflow or Apache Flink to perform data transformations.

Pub/Sub offers **Single Message Transforms (SMTs)** to simplify data transformations for streaming pipelines. SMTs enable lightweight modifications to message data and attributes directly within Pub/Sub. SMTs eliminate the need for additional data processing steps or separate data transformation products.

When an SMT is run, it takes the Pub/Sub message as input, including the message data and attributes. The output is a transformed Pub/Sub message, with modifications to the data or attributes. SMTs are integrated into the Pub/Sub API, so you can manage them as part of your topic or subscription configurations.

100

You can use Pub/Sub SMTs to make data transformations such as the following:

- Remove personally identifiable information (PII), such as full names and addresses, to protect customer privacy.
- Retain only recommendation-relevant events, such as product views and purchases, and discard others, such as customer profile changes.
- Ensure all timestamps, currency values, and product IDs adhere to a consistent format and type compatible with the recommendation system.
- Generate new data fields from raw data, such as shopping cart total value or product page dwell time.
- Add inferences from Vertex AI models to event data, such as classifications, predictions, sentiments, or embeddings.

101

Topic

A topic resource.

Fields	
name	string

Required. Identifier. Name of the topic. Format is `projects/{project}/topics/{topic}`.

102

¹⁰⁰ <https://docs.cloud.google.com/pubsub/docs/smts/smts-overview>

¹⁰¹ *Id.*

¹⁰² <https://docs.cloud.google.com/pubsub/docs/reference/rpc/google.pubsub.v1#topic>

“obtaining, by the computing device, context information in accordance with the indication;”

119. Google’s Kafka Service performs the step of obtaining, by the computing device, context information in accordance with the indication.

120. For example, as explained above, Kafka obtains context information (e.g., topic name, partition index) in accordance with the indication. One example of that is shown below:

Requests:

```
Produce Request (Version: 3) => transactional_id acks timeout_ms [topic_data]
transactional_id => NULLABLE_STRING
acks => INT16
timeout_ms => INT32
topic_data => name [partition_data]
name => STRING
partition_data => index records
index => INT32
records => RECORDS
```

Request header version: 1

FIELD	DESCRIPTION
transactional_id	The transactional ID, or null if the producer is not transactional.
acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete. Allowed values: 0 for no acknowledgments, 1 for only the leader and -1 for the full ISR.
timeout_ms	The timeout to await a response in milliseconds.
topic_data	Each topic to produce to.
name	The topic name.
partition_data	Each partition to produce to.
index	The partition index.
records	The record data to be produced.

```
Produce Request (Version: 4) => transactional_id acks timeout_ms [topic_data]
transactional_id => NULLABLE_STRING
acks => INT16
timeout_ms => INT32
topic_data => name [partition_data]
name => STRING
partition_data => index records
index => INT32
records => RECORDS
```

103

121. Google’s Pub/Sub performs the step of obtaining, by the computing device, context information in accordance with the indication. For example, as explained above, when it is selectively indicated that context information is to be obtained, a SMT will obtain the context information and append it to the message:

¹⁰³ <https://kafka.apache.org/42/design/protocol/#the-messages>

Sample message flow for SMTs

The image shows an example Pub/Sub system with SMTs applied at both the topic and subscription levels.

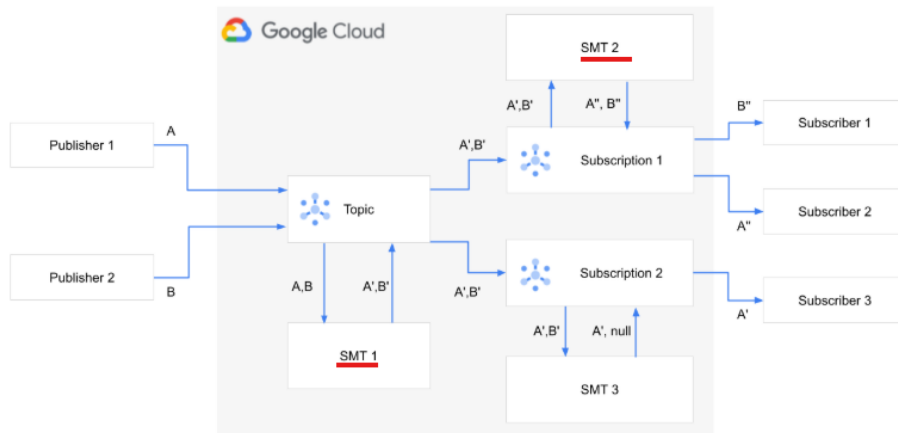


Figure 1 How messages are transformed with SMTs.

104

JavaScriptUDF

User-defined JavaScript function that can transform or filter a Pub/Sub message.

Fields	
function_name	string Required. Name of the JavaScript function that should applied to Pub/Sub messages.
code	string Required. JavaScript code that contains a function function_name with the below signature:

```

/**
 * Transforms a Pub/Sub message.
 *
 * @return {(Object<string, (string | Object<string, string>)>|null)} - To
 * filter a message, return 'null'. To transform a message return a map
 * with the following keys:
 * - (required) 'data' : {string}
 * - (optional) 'attributes' : {Object<string, string>}
 * Returning empty 'attributes' will remove all attributes from the
 * message.
 *
 * @param {(Object<string, (string | Object<string, string>)>| Pub/Sub
 * message. Keys:
 * - (required) 'data' : {string}
 * - (required) 'attributes' : {Object<string, string>}
 *
 * @param {Object<string, any>} metadata - Pub/Sub message metadata.
 * Keys:
 * - (optional) 'message_id' : {string}
 * - (optional) 'publish_time' : {string} YYYY-MM-DDTHH:MM:SSZ format
 * - (optional) 'ordering_key' : {string}
 */
function <function_name>(message, metadata) {

```

105

¹⁰⁴ <https://docs.cloud.google.com/pubsub/docs/smts/smts-overview>

¹⁰⁵

<https://docs.cloud.google.com/pubsub/docs/reference/rpc/google.pubsub.v1#google.pubsub.v1.JavaScriptUDF>

“and adding, by the computing device, the obtained context information to the outgoing message such that a response, received from the destination node, to the outgoing message contains the obtained context information.”]

122. Google’s Kafka Service performs the step of adding, by the computing device, the obtained context information to the outgoing message such that a response, received from the destination node, to the outgoing message contains the obtained context information.

123. For example, as explained above, Google’s Kafka Service appends the context information (e.g., topic name and partition index) to the outgoing message. A response received from the destination node (e.g., the Kafka broker) contains the obtained context information. The below is an example of both the request and response information:

Request header version: 2	
FIELD	DESCRIPTION
transactional_id	The transactional ID, or null if the producer is not transactional.
acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete. Allowed values: 0 for no acknowledgments, 1 for only the leader and -1 for the full ISR.
timeout_ms	The timeout to await a response in milliseconds.
topic_data	Each topic to produce to.
name	The topic name.
partition_data	Each partition to produce to.
index	The partition index.
records	The record data to be produced.

Produce Request (Version: 10) => transactional_id acks timeout_ms [topic_data]	
transactional_id =>	COMPACT_NULLABLE_STRING
acks =>	INT16
timeout_ms =>	INT32
topic_data =>	name [partition_data]
name =>	COMPACT_STRING
partition_data =>	index records
index =>	INT32
records =>	COMPACT_RECORDS

106

¹⁰⁶ <https://kafka.apache.org/protocol>

FIELD	DESCRIPTION
responses	Each produce response.
<u>name</u>	The topic name.
partition_responses	Each partition that we produced to within the topic.
<u>index</u>	The partition index.
error_code	The error code, or 0 if there was no error.
base_offset	The base offset.
log_append_time_ms	The timestamp returned by broker after appending the messages. If CreateTime is used for the topic, the timestamp will be -1. If LogAppendTime is used for the topic, the timestamp will be the broker local time when the messages are appended.
log_start_offset	The log start offset.
record_errors	The batch indices of records that caused the batch to be dropped.
batch_index	The batch index of the record that caused the batch to be dropped.
batch_index_error_message	The error message of the record that caused the batch to be dropped.
error_message	The global error message summarizing the common root cause of the records that caused the batch to be dropped.
throttle_time_ms	The duration in milliseconds for which the request was throttled due to a quota violation, or zero if the request did not violate any quota.


```

Produce Response (Version: 10) => [responses] throttle_time_ms [node_endpoints]<tag: 0>
responses => name [partition_responses]
name => COMPACT_STRING
partition_responses => index error_code base_offset log_append_time_ms log_start_offset [record_errors] error_message current_leader<tag: 0>
index => INT32
error_code => INT16
base_offset => INT64
log_append_time_ms => INT64
log_start_offset => INT64
record_errors => batch_index batch_index_error_message
batch_index => INT32
batch_index_error_message => COMPACT_NULLABLE_STRING
error_message => COMPACT_NULLABLE_STRING
current_leader<tag: 0> => leader_id leader_epoch
leader_id => INT32
leader_epoch => INT32
throttle_time_ms => INT32
node_endpoints<tag: 0> => node_id host port rack
node_id => INT32
host => COMPACT_STRING
port => INT32
rack => COMPACT_NULLABLE_STRING

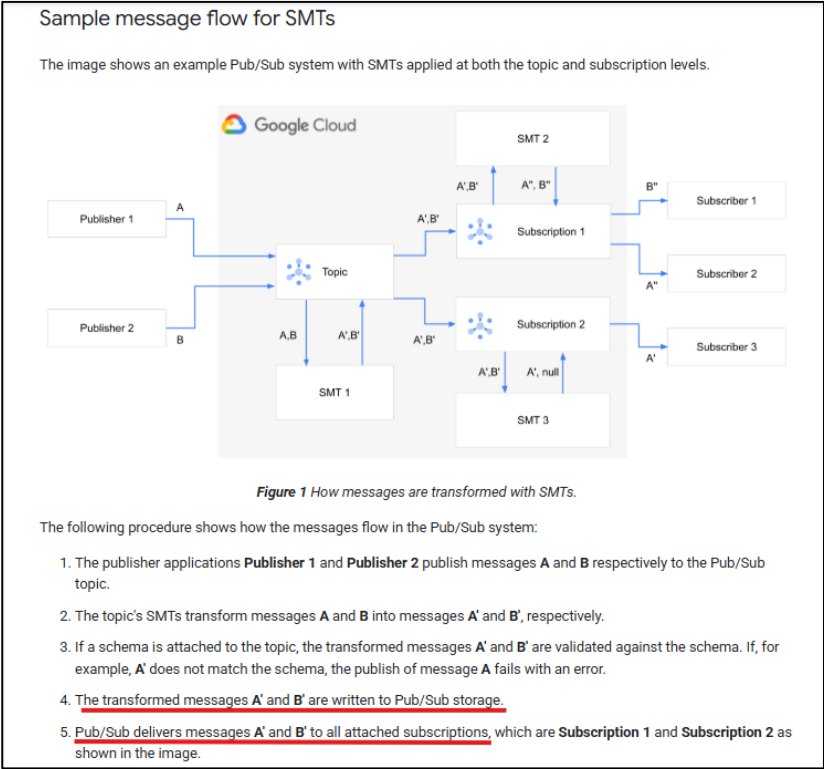
```

107

124. Google's Pub/Sub also performs the step of adding, by the computing device, the obtained context information to the outgoing message such that a response, received from the destination node, to the outgoing message contains the obtained context information. As explained above, Pub/Sub appends the obtained context information to the outgoing message. The response to the message contains the obtained context information, as seen below:

¹⁰⁷ *Id.*

DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT



108

ReceivedMessage

A message and its corresponding acknowledgment ID.

Fields	
ack_id	string Optional. This ID can be used to acknowledge the received message.
message	PubsubMessage Optional. The message.
delivery_attempt	int32 Optional. The approximate number of times that Pub/Sub has attempted to deliver the associated message to a subscriber. More precisely, this is 1 + (number of NACKs) + (number of ack_deadline exceeds) for this message. A NACK is any call to ModifyAckDeadline with a 0 deadline. An ack_deadline exceeds event is whenever a message is not acknowledged within ack_deadline. Note that ack_deadline is initially Subscription.ackDeadlineSeconds, but may get extended automatically by the client library. Upon the first delivery of a given message, delivery_attempt will have a value of 1. The value is calculated at best effort and is approximate. If a DeadLetterPolicy is not set on the subscription, this will be 0.

109

¹⁰⁸ <https://docs.cloud.google.com/pubsub/docs/smts/smts-overview>

¹⁰⁹ <https://docs.cloud.google.com/pubsub/docs/reference/rpc/google.pubsub.v1#google.pubsub.v1.PubsubMessage>

PubsubMessage

A message that is published by publishers and consumed by subscribers. The message must contain either a non-empty data field or at least one attribute. Note that client libraries represent this object differently depending on the language. See the corresponding [client library documentation](#) for more information. See [quotas and limits](#) for more information about message limits.

Fields	
data	bytes Optional. The message data field. If this field is empty, the message must contain at least one attribute.
<u>attributes</u>	map<string, string> Optional. Attributes for this message. If this field is empty, the message must contain non-empty data. This can be used to filter messages on the subscription.
message_id	string ID of this message, assigned by the server when the message is published. Guaranteed to be unique within the topic. This value may be read by a subscriber that receives a PubsubMessage via a Pull call or a push delivery. It must not be populated by the publisher in a Publish call.
publish_time	Timestamp The time at which the message was published, populated by the server when it receives the Publish call. It must not be populated by the publisher in a Publish call.
ordering_key	string Optional. If non-empty, identifies related messages for which publish order should be respected. If a Subscription has enable_message_ordering set to true, messages published with the same non-empty ordering_key value will be delivered to subscribers in the order in which they are received by the Pub/Sub system. All PubsubMessages published in a given PublishRequest must specify the same ordering_key value. For more information, see ordering messages .

110

125. Google is also inducing infringement of the '738 Patent, in violation of 35 U.S.C. § 271(b), including by knowingly encouraging or aiding third parties to make, use, sell, or offer to sell the Kafka Service and/or Pub/Sub Service and/or Products using these Services such as Google Ads, Search, and Gmail in the United States, or to import the Kafka Service and/or Pub/Sub Service and/or Products using these Services such as Google Ads, Search, and Gmail into the United States, and/or in violation of 35 U.S.C. § 271(f)(1), by supplying or causing to be supplied the Kafka Service and/or Pub/Sub Service and/or Products using these Services such as Google Ads, Search, and Gmail in or from the United States, without license or authority from Defendants, with

¹¹⁰ *Id.*

1 knowledge of or willful blindness to the fact that Google's actions will induce others to directly
2 infringe the '738 Patent. Google actively encourages, directs, and controls third parties, including
3 its end users, to make and/or use the Kafka Service and/or Pub/Sub Service and/or Products using
4 these Services such as Google Ads, Search, and Gmail in an infringing manner, including through
5 Google's website, advertisements, documentation/instructional material, customer support,
6 application and feature descriptions, and in-application instructions. Google has intentionally aided
7 and encouraged third parties, including its end users, to use the Kafka Service and/or Pub/Sub
8 Service and/or Products using these Services such as Google Ads, Search, and Gmail in the United
9 States in a manner that it knows would infringe or have a high probability of infringing the '738
10 Patent, with the specific intent that those performing the acts infringe the '738 Patent.
11

12 126. Google is contributorily infringing the '738 Patent in violation of 35 U.S.C. § 271(c),
13 by selling, offering to sell, and/or importing into the United States, and/or in violation of 35 U.S.C.
14 § 271(f)(2), by supplying or causing to be supplied in or from the United States, the Kafka Service
15 and/or Pub/Sub Service and/or Products using these Services such as Google Ads, Search, and
16 Gmail for use in infringing the '738 Patent, constituting a material part of the invention, knowing
17 the Kafka Service and/or Pub/Sub Service and/or Products using these Services such as Google Ads,
18 Search, and Gmail to be especially made or especially adapted for use in infringing the '738 Patent.
19 The Kafka Service and/or Pub/Sub Service and/or Products using these Services such as Google
20 Ads, Search, and Gmail are not staple articles or commodities of commerce suitable for substantial
21 non-infringing use.
22

23
24 127. As a result of Google's infringement of the '738 Patent, Defendants have been
25 damaged. In addition, Google's infringing acts and practices have caused and are causing immediate
26 and irreparable harm to Defendants. Defendants are entitled to recover for damages sustained as a
27
28

1 result of Google's wrongful acts in an amount yet to be determined and to receive such other and
 2 further relief, including equitable relief, as this Court deems just and proper.

3 128. Defendants are further informed, and on this basis allege, that Google's infringement
 4 of the '738 Patent has been and continues to be deliberate and willful, and, therefore, this is an
 5 exceptional case warranting an award of enhanced damages for up to three times the actual damages
 6 awarded and attorney's fees to Defendants pursuant to 35 U.S.C. §§ 284-285. As noted above,
 7 Google has had knowledge of the '738 Patent and its infringement thereof, including through its
 8 awareness of Defendants' action against SAP America for its use of Apache Kafka, *Valtrus*
 9 *Innovations Ltd. v. SAP America, Inc.*, No. 2:24-cv-00021 (E.D. Tex. Jan. 15, 2024), and yet Google
 10 has deliberately continued to infringe in a wanton, malicious, and egregious manner, with reckless
 11 disregard for Defendants' patent rights.
 12

13 **PRAYER FOR RELIEF**

14 WHEREFORE, Defendants request that the relief sought by Google is denied, and pray for
 15 judgment against Google as follows:
 16

- 17 A. That Google has infringed each of the Asserted Patents;
- 18 B. That Google has willfully infringed one or more of the Asserted Patents;
- 19 C. That Google pay Defendants damages adequate to compensate Defendants for
 20 Google's past infringement of each of the Asserted Patents, together with interests and costs
 21 under 35 U.S.C. § 284;
- 22 D. That Google be ordered to pay pre-judgment and post-judgment interest to
 23 Defendants on the damages assessed;
- 24 E. That Google pay enhanced damages to Defendants pursuant to 35 U.S.C. § 284;
- 25 F. That Google pay supplemental damages to Defendants, including interest, with
 26 accounting, as needed;
 27
 28

DEFENDANTS' ANSWER, AFFIRMATIVE
 DEFENSES, AND COUNTERCLAIMS TO
 PLAINTIFF'S COMPLAINT

1 G. That this is an exceptional case under 35 U.S.C. § 285 and that Google pay
2 Defendants' attorneys' fees and costs in this action; and

3 H. That Defendants be awarded such other and further relief, including equitable relief,
4 as this Court deems just and proper.

5
6 **DEMAND FOR JURY TRIAL**

7 Pursuant to Federal Rule of Civil Procedure 38(b), Defendants hereby demand a trial by jury
8 on all issues triable to a jury.

1
2 Dated: April 16, 2026

Respectfully submitted,

3
4 By: /s/ Jason G. Sheasby

5 Jason G. Sheasby (SBN 205455)
jsheasby@irell.com
6 Andrew Strabone (SBN 301659)
astrabone@irell.com
7 Lucas S. Oxenford (SBN 328152)
loxenford@irell.com
8 Steven Marshack (SBN 364249)
smarshack@irell.com
9 IRELL & MANELLA LLP
1800 Avenue of the Stars, Ste. 900
10 Los Angeles, CA 90067
310-277-1010

11 Anthony Q. Rowles (SBN 301209)
trowles@irell.com
12 IRELL & MANELLA LLP
840 Newport Center Dr #400
13 Newport Beach, CA 92660
949-760-0991

14 *Counsel for Defendants*
15 *Valtrus Innovations Limited and Key Patent*
16 *Innovations Limited*

17
18
19
20
21
22
23
24
25
26
27
28 DEFENDANTS' ANSWER, AFFIRMATIVE
DEFENSES, AND COUNTERCLAIMS TO
PLAINTIFF'S COMPLAINT

EXHIBIT 1

(12) **United States Patent**
Gualdi et al.

(10) **Patent No.: US 7,057,509 B2**
(45) **Date of Patent: Jun. 6, 2006**

(54) **MONITORING AN OBJECT WITH IDENTIFICATION DATA AND TRACKING DATA**

(75) Inventors: **Giovanni Gualdi**, Palo Alto, CA (US);
Cyril Brignone, Mountain View, CA (US); **Salil Pradhan**, San Jose, CA (US)

(73) Assignee: **Hewlett-Packard Development Company, L.P.**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 103 days.

(21) Appl. No.: **10/881,975**

(22) Filed: **Jun. 29, 2004**

(65) **Prior Publication Data**
US 2005/0285733 A1 Dec. 29, 2005

(51) **Int. Cl.**
G08B 1/08 (2006.01)

(52) **U.S. Cl.** **340/539.13**; 340/572.1;
340/568.1; 340/539.1; 340/5.1; 340/5.2; 340/933;
340/937

(58) **Field of Classification Search** 340/572.1,
340/568.1, 5.1, 5.2, 539.1, 539.13, 933, 937
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,526,158 B1	2/2003	Goldberg	
6,816,720 B1 *	11/2004	Hussain et al.	455/404.2
2004/0143602 A1 *	7/2004	Ruiz et al.	707/104.1
2005/0088320 A1 *	4/2005	Kovach	340/933

FOREIGN PATENT DOCUMENTS

EP	1 578 130	9/2005
WO	WO 01/46923	6/2001
WO	WO 2004/051590	6/2004
WO	WO 2004/051985	6/2004

* cited by examiner

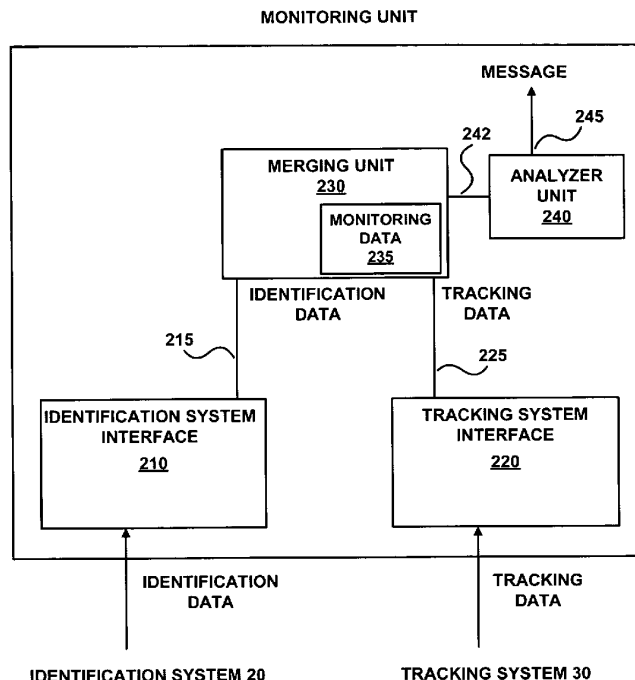
Primary Examiner—Julie Bichngoc Lieu

(57) **ABSTRACT**

An object is monitored with identification data and tracking data. In an embodiment, a monitoring apparatus is utilized to monitor the object. The monitoring apparatus has a first interface for receiving identification data from an identification system. Moreover, the monitoring apparatus includes a second interface for receiving tracking data from a tracking system. Additionally, the monitoring apparatus further includes a merging unit for merging and storing the identification data and the tracking data of each monitored object to form monitoring data for each monitored object.

30 Claims, 11 Drawing Sheets

10



U.S. Patent

Jun. 6, 2006

Sheet 1 of 11

US 7,057,509 B2

100

MONITORING SYSTEM

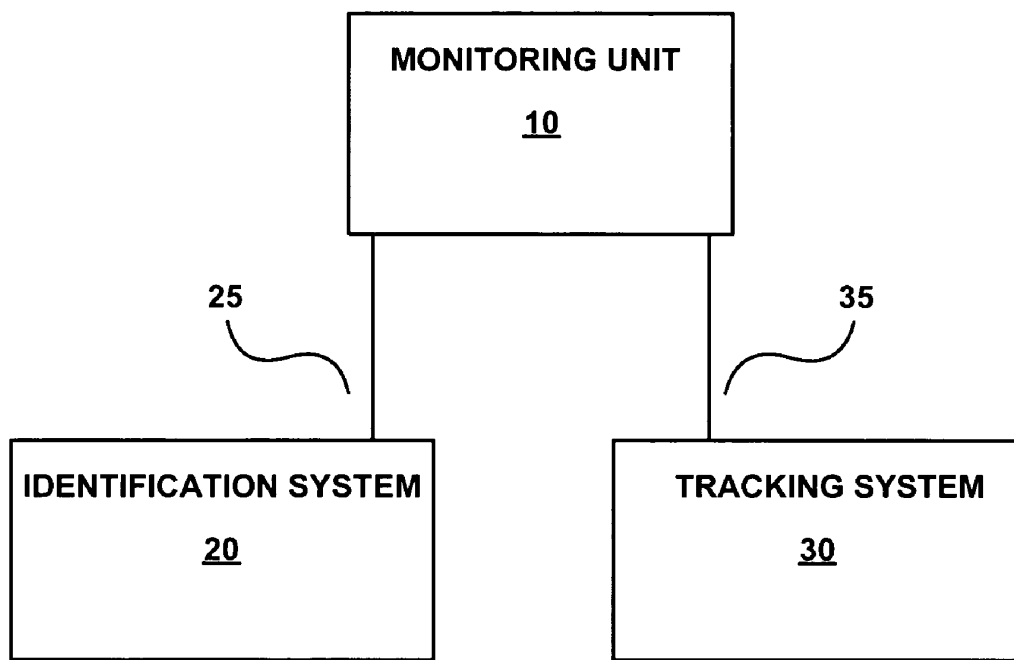


FIG. 1

U.S. Patent

Jun. 6, 2006

Sheet 2 of 11

US 7,057,509 B2

10

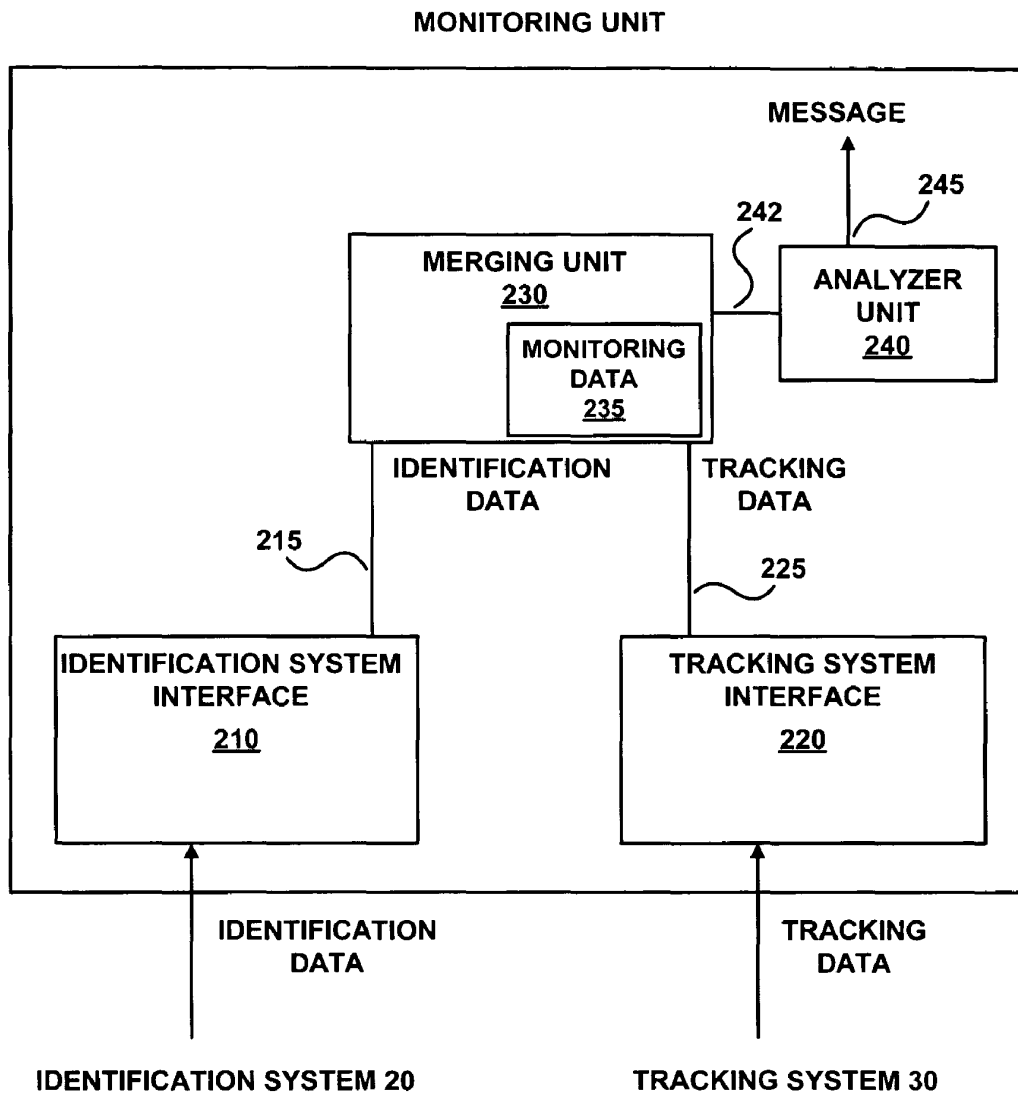


FIG. 2

U.S. Patent

Jun. 6, 2006

Sheet 3 of 11

US 7,057,509 B2

235

DATA STRUCTURE FOR MONITORING DATA

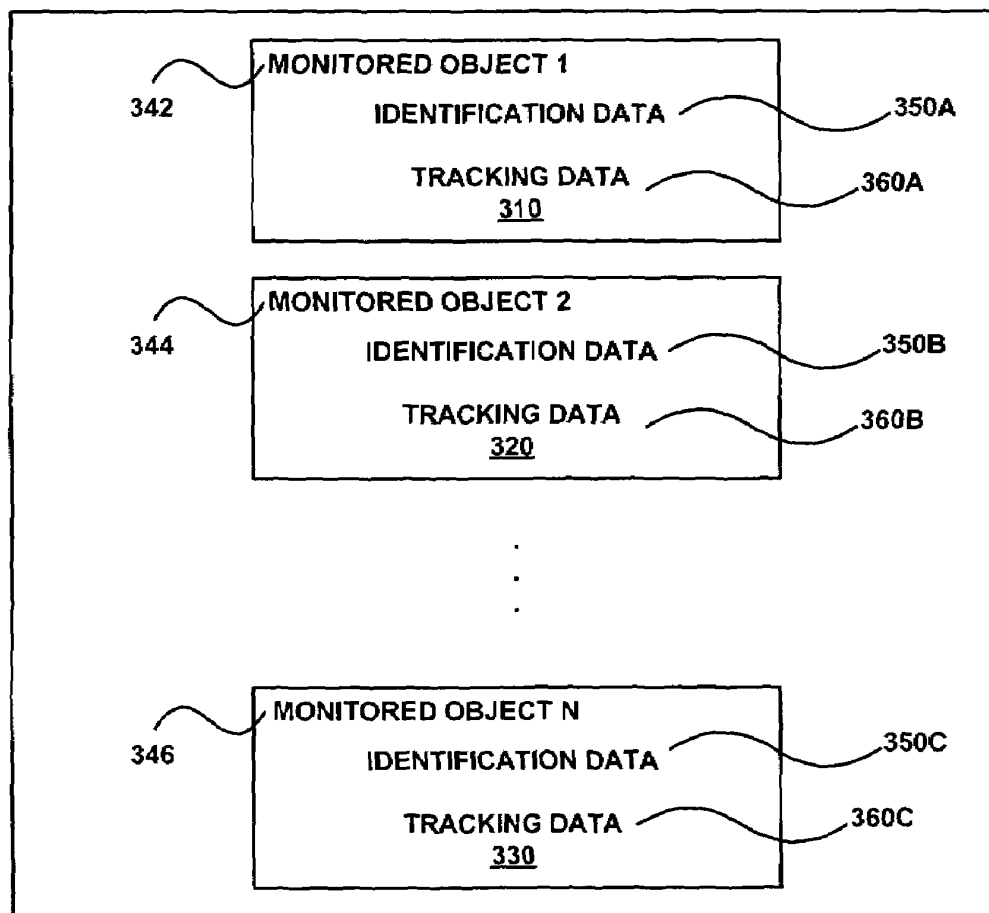


FIG. 3

U.S. Patent

Jun. 6, 2006

Sheet 4 of 11

US 7,057,509 B2

400

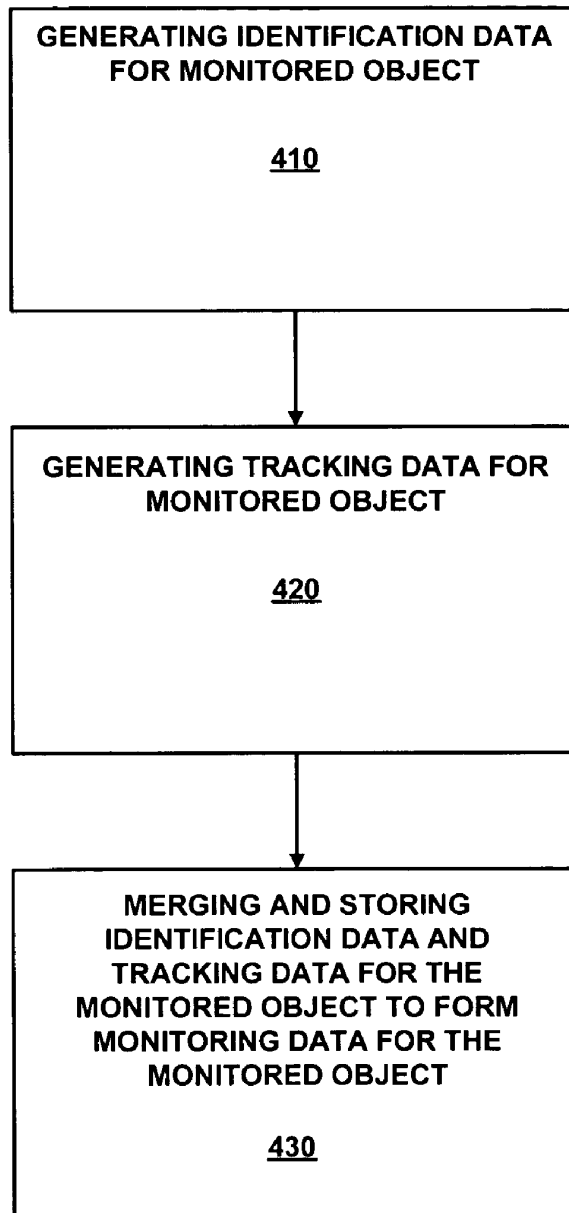


FIG. 4

U.S. Patent

Jun. 6, 2006

Sheet 5 of 11

US 7,057,509 B2

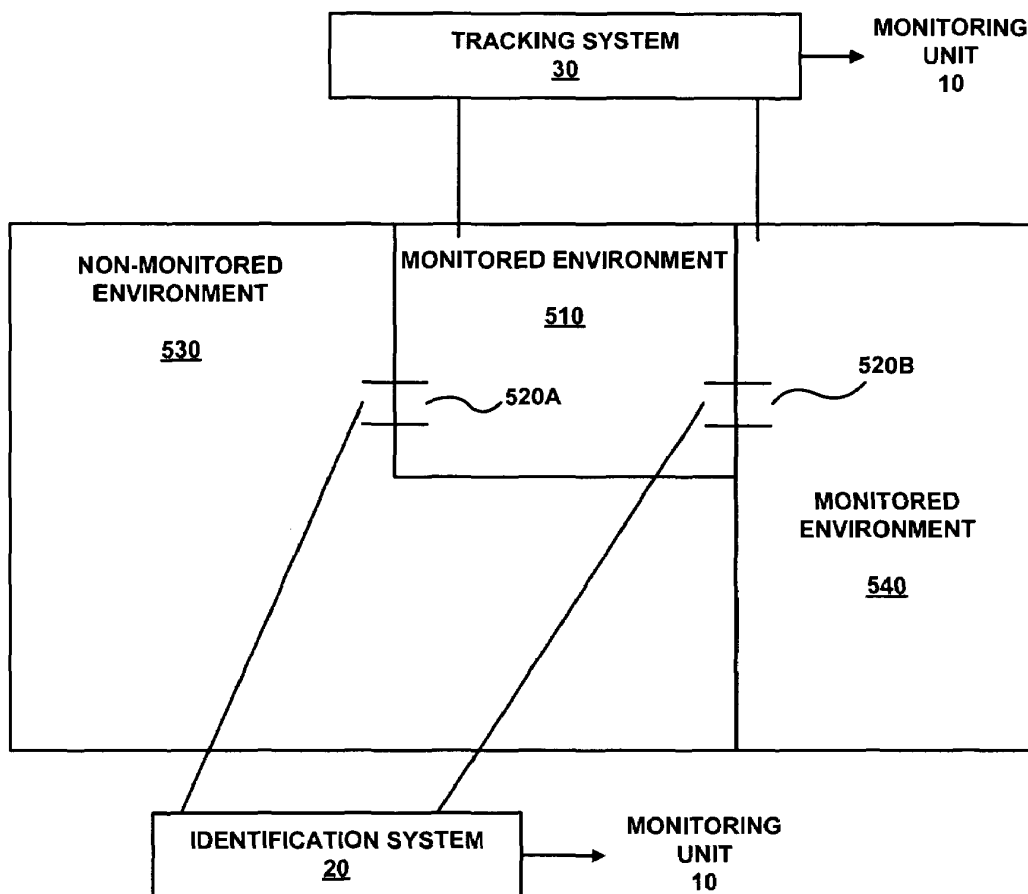


FIG. 5

U.S. Patent

Jun. 6, 2006

Sheet 6 of 11

US 7,057,509 B2

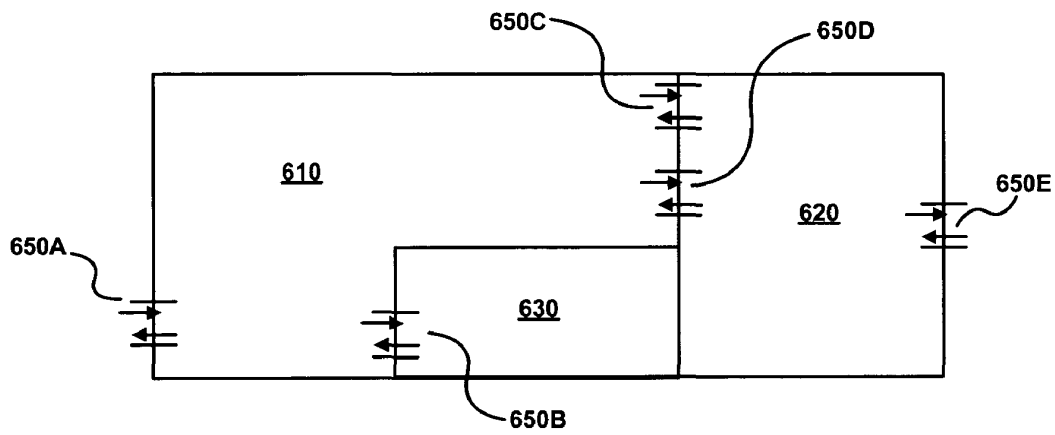


FIG. 6

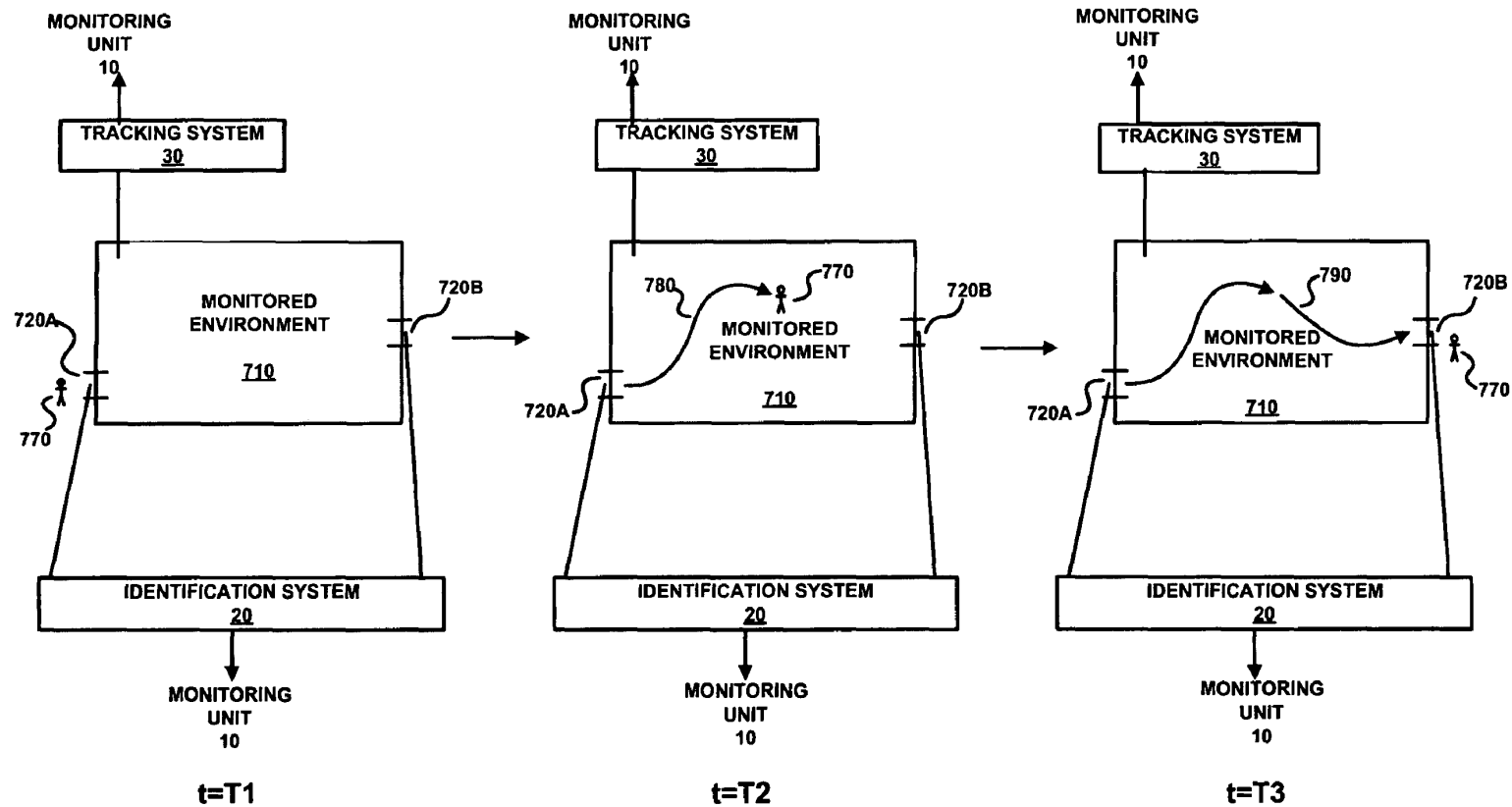


FIG. 7

U.S. Patent

Jun. 6, 2006

Sheet 8 of 11

US 7,057,509 B2

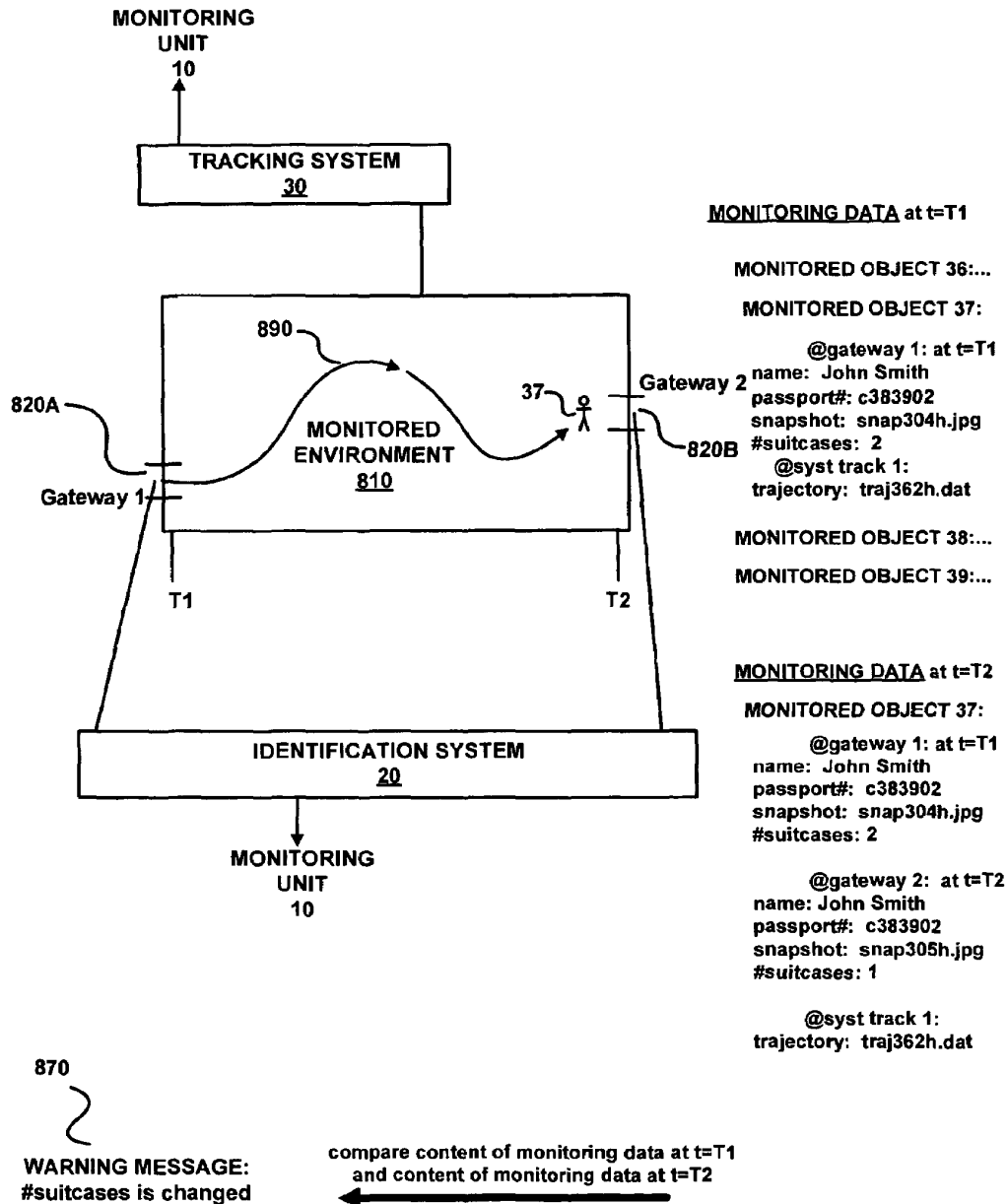


FIG. 8

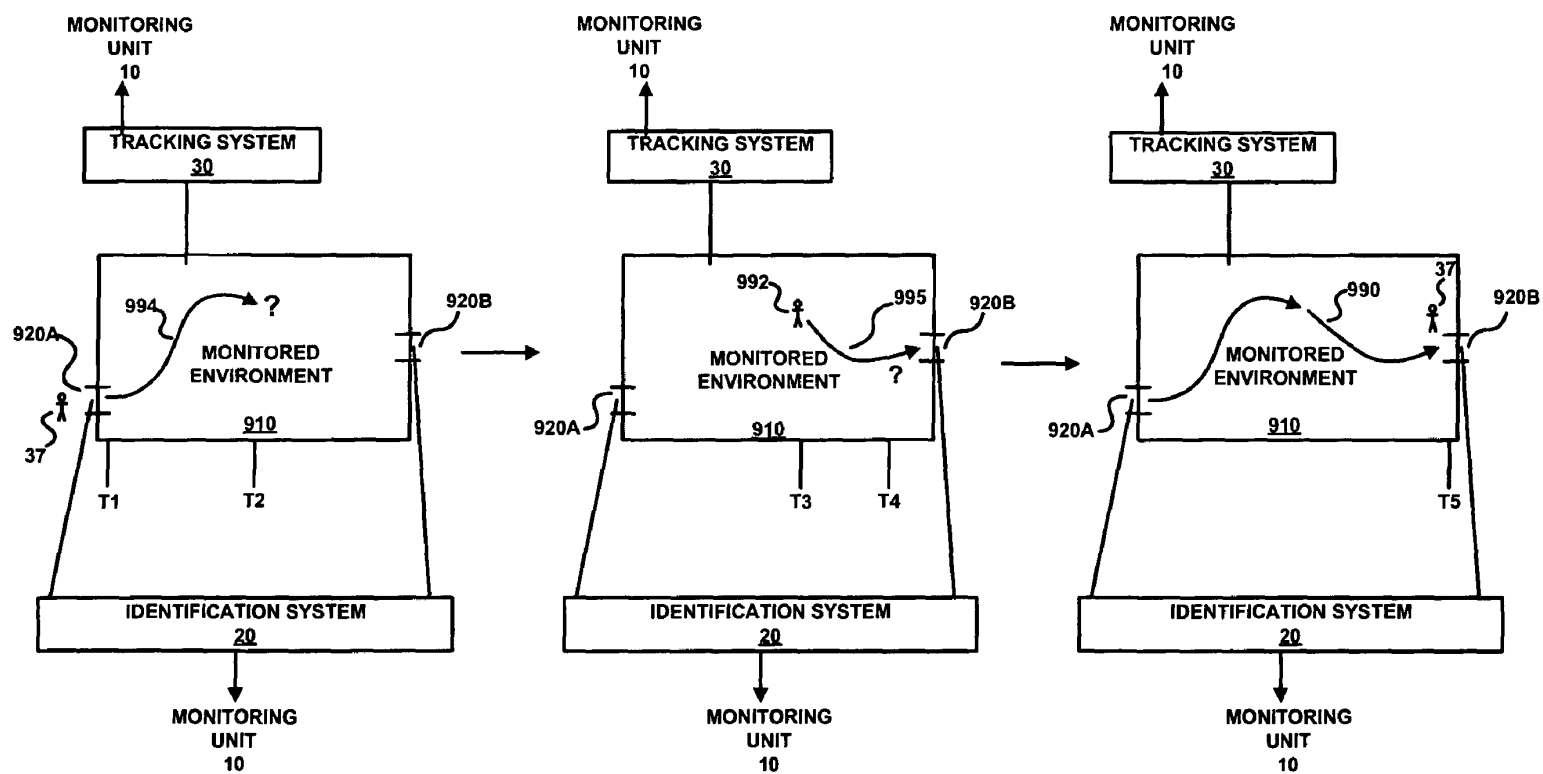


FIG. 9

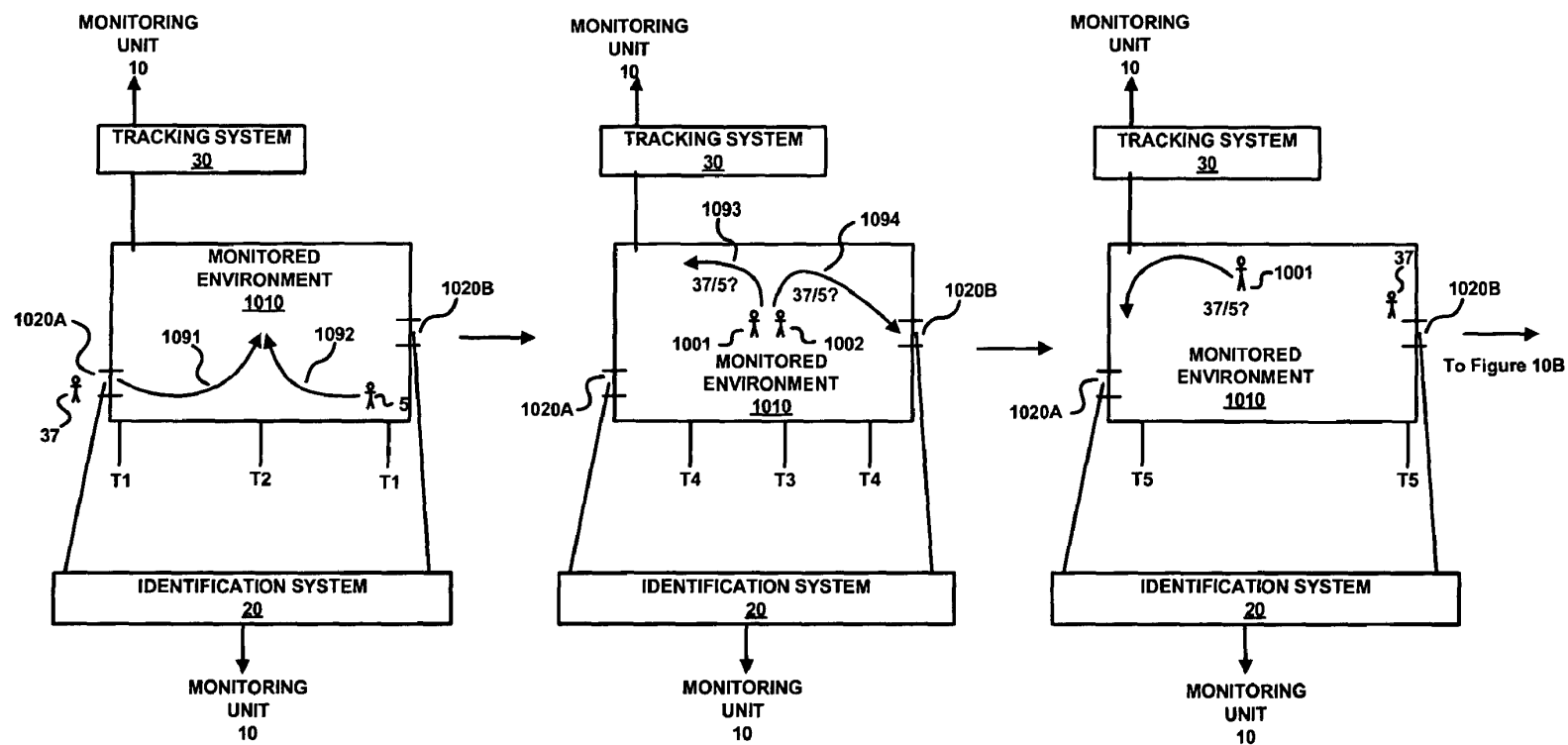


FIG. 10A

U.S. Patent

Jun. 6, 2006

Sheet 11 of 11

US 7,057,509 B2

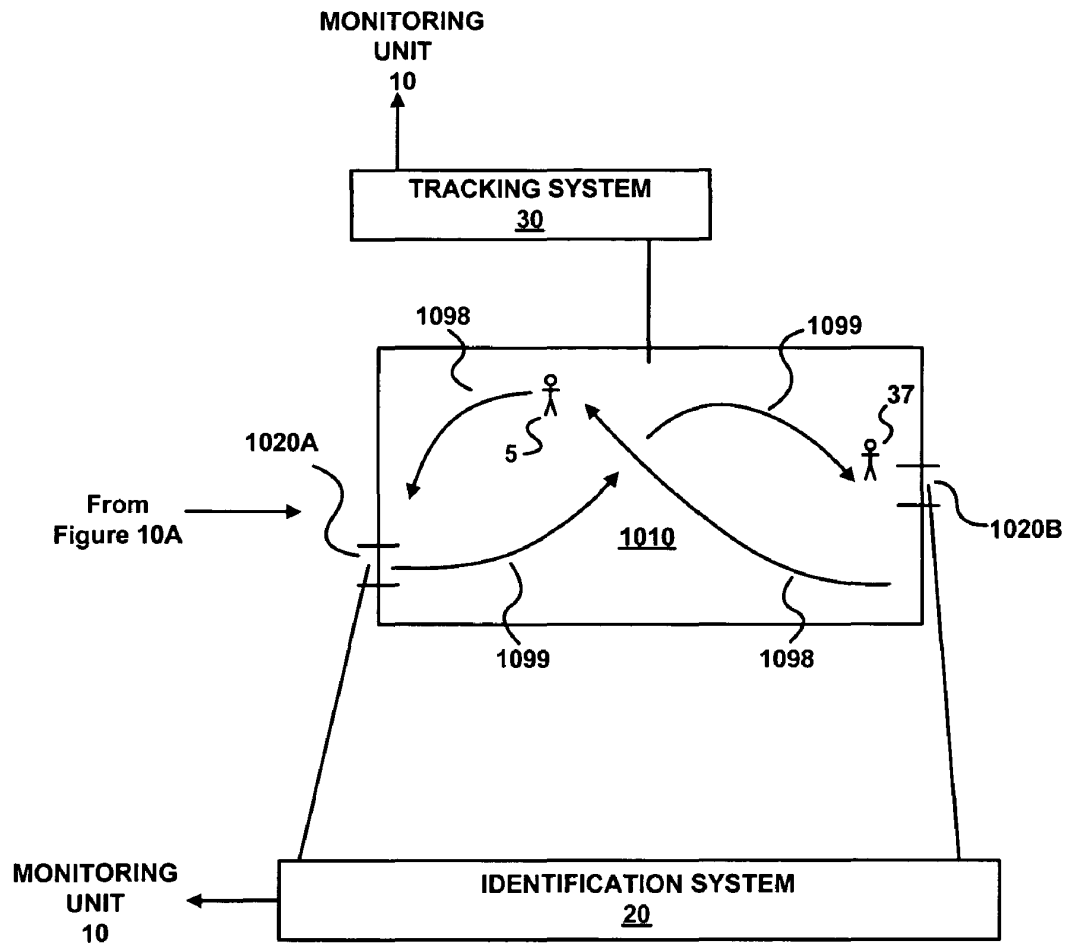


FIG. 10B

US 7,057,509 B2

1

MONITORING AN OBJECT WITH IDENTIFICATION DATA AND TRACKING DATA

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to methods and systems for monitoring an object. More particularly, the present invention relates to monitoring an object with identification data and tracking data.

2. Related Art

In general, monitoring systems have been developed for specific applications. These monitoring systems attempt to let a user be aware of what is happening in the monitored environment and what objects are involved. Examples of objects include persons, vehicles, boxes, pallets, carts, and any other kind of object.

Typically, these monitoring systems focus on either identifying the objects or tracking what is happening in the monitored environment.

Monitoring systems that focus on identifying the objects can provide high accuracy in identifying the objects. However, these monitoring systems typically are deficient in several ways. First, the high accuracy in identifying objects is spatially limited. That is, the objects have to be within a particular distance of the identification sensors of the monitoring system to maintain the high accuracy. Beyond the particular distance, the accuracy can drop significantly. Secondly, since the focus is on identifying the objects, these monitoring systems typically lack or fail to provide sufficiently reliable tracking sensors to track the activity of the objects outside the scope of the identification sensors.

Alternatively, monitoring systems that focus on tracking what is happening in the monitored environment can track the activity of the objects in the monitored environment, where the monitored environment typically can be any desired size or shape. For example, the monitored environment can be small or large in size. Unfortunately, the accuracy of these monitoring systems typically decreases as the size of the monitored environment is increased. Moreover, these monitoring systems tend to assign tracking identifiers to each monitored object. These tracking identifiers usually are unrelated to the real identity of the monitored object.

A monitoring system capable of providing automated monitoring with the desired level of accuracy and flexibility is needed.

SUMMARY OF THE INVENTION

An object is monitored with identification data and tracking data. In an embodiment, a monitoring apparatus is utilized to monitor the object. The monitoring apparatus has a first interface for receiving identification data from an identification system. Moreover, the monitoring apparatus includes a second interface for receiving tracking data from a tracking system. Additionally, the monitoring apparatus further includes a merging unit for merging and storing the identification data and the tracking data of each monitored object to form monitoring data for each monitored object.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments

2

of the invention and, together with the description, serve to explain the principles of the present invention.

FIG. 1 illustrates a monitoring system in accordance with an embodiment of the present invention.

FIG. 2 illustrates a block diagram of the monitoring unit of FIG. 1 in accordance with an embodiment of the present invention.

FIG. 3 illustrates data structure of monitoring data in accordance with an embodiment of the present invention.

FIG. 4 illustrates a flow chart showing a method of monitoring an object in accordance with an embodiment of the present invention.

FIG. 5 illustrates operation of identification system and tracking system in accordance with an embodiment of the present invention.

FIG. 6 illustrates monitored environments in accordance with an embodiment of the present invention.

FIG. 7 illustrates operation of the monitoring system of FIG. 1 in accordance with an embodiment of the present invention.

FIG. 8 illustrates comparison functionality of the monitoring system of FIG. 1 in accordance with an embodiment of the present invention.

FIG. 9 illustrates a first error recovery functionality of the monitoring system of FIG. 1 in accordance with an embodiment of the present invention.

FIGS. 10A and 10B illustrate a second error recovery functionality of the monitoring system of FIG. 1 in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Reference will now be made in detail to embodiments of the present invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with these embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention.

In an embodiment of the present invention, a monitoring system merges a tracking system and an identification system to obtain the desired level of accuracy and flexibility. Although the description will focus on non-invasive tracking systems, it should be understood that the present invention is equally applicable to invasive tracking systems. A non-invasive tracking system is configured to provide tracking functionality independently of the tracked object. For example, a tracking system that uses cameras as sensors to track the object is a type of non-invasive tracking system. An invasive tracking system is configured to provide tracking functionality dependent on something in the possession of the tracked object. For example, a tracking system that uses radio frequency transmitters coupled to the object and radio frequency sensors to track the object is a type of invasive tracking system.

FIG. 1 illustrates a monitoring system 100 in accordance with an embodiment of the present invention. As depicted in FIG. 1, the monitoring system 100 includes a monitoring unit 10, an identification system 20, and a tracking system 30. The identification system 20 and the tracking system 30 are coupled to the monitoring unit 10 via connections 25 and

US 7,057,509 B2

3

35, respectively. The monitoring system 100 allows users to be aware of what is happening in the monitored environment (or event) and who are the objects involved.

The monitoring system 100 merges tracking and identification functionalities from the identification system 20 and the tracking system 30 to exhibit several beneficial characteristics. The monitoring system 100 is able to monitor objects (e.g., persons, vehicles, boxes, pallets, carts, or any other kind of object) in a monitored environment that can range in size from small to large (e.g., room, aisle, floor, building, parking lot, etc.). Additionally, the monitoring system 100 is aware of the position of the object at desired time intervals via the tracking sensors of the tracking system 30. In an embodiment, the tracking system 30 uses non-invasive sensors, reducing the invasiveness of monitoring system 100 on the monitored environment. Moreover, monitoring system 100 is able to merge and store the identification data from the identification system 20 and the tracking data from the tracking system 30 for each monitored object to form monitoring data for each monitored object, enabling analysis and queries of this monitoring data.

The merging of tracking and identification functionalities makes the monitoring system 100 suitable for automated monitoring and non-automated monitoring applications. As will be described below, a level of accuracy in the tracking functionality suitable for automated monitoring applications is achieved by utilizing the accuracy of the identification functionality. Also, a desired level of detail in the description of the activity (or events) and the monitored objects involved is achieved by utilizing the tracking functionality and the identification functionality. For example, a person can be described by unique meaningless code, name, employee number, passport number, height, the shape of the iris, or any combination thereof. Moreover, the monitoring system 100 is sufficiently flexible to allow a variable level of human interaction and automatic functionality, as needed by the specific application.

Referring to FIG. 1, the tracking system 30 is able to determine/detect the presence of objects in the monitored environment. Moreover, the tracking system 30 associates a unique tracking identifier with the object. Furthermore, the tracking system 30 tracks the position of the object at desired time intervals to obtain a trajectory (e.g., in a coordinate system) for the object within the monitored environment. The unique tracking identifier distinguishes the objects. Maintaining the correct association between the unique tracking identifier and the object affects the accuracy of the tracking system 30. Also, the tracking system 30 can be several tracking subsystems that are functionally integrated or functionally independent of each other.

The accuracy of the tracking system 30 widely depends on the number, the type, and quality of the tracking sensors used. The tracking system 30 is not limited to any particular type of tracking sensor. In an embodiment, the tracking system 30 uses tracking sensors that are cameras, reducing the invasiveness of the monitoring system 100 on the monitored environment. Examples of cameras suitable for the tracking system 30 include color cameras, black-and-white cameras, infrared (IR) cameras, and range cameras. Cameras are considered non-invasive tracking sensors since the objects do not need to be equipped with anything specific to be tracked.

Generally, the tracking sensors provide a detailed level of description of the tracked objects. This description of the tracked object, the trajectory of the tracked object, and the unique tracking identifier are examples of tracking data

4

generated by the tracking system 30. This tracking data is sent to the monitoring unit 10 and processed as described below.

Continuing with FIG. 1, the identification system 20 identifies the object by matching the object with one of the plurality of identities stored by the identification system 20. Generally, the level of accuracy of the identification system 20 is higher than the level of accuracy of the tracking system 30. This is possible because the identification system 20 is local. That is, the object is identified at a particular location. Moreover, the identification system 20 can be implemented as several identification subsystems that are functionally integrated or functionally independent of each other. Hence, the identification system 20 can utilize automated identification systems, human-assisted identification systems, or a combination of both. A typical human-assisted identification system is given by a police officer that checks the passport number from the people passing through a security point. An automatic identification system can be based on pattern recognition (e.g., face recognition, iris recognition, fingerprint recognition, voice recognition, etc.). As another example, the automatic identification system can be based on RFID (radio frequency identification) technology. This type of automatic identification system allows wireless recovery of the numeric code on an ID-tag equipped object, using a RFID reader.

Further, the identification system 20 can retrieve the identity of the object and gathers additional data about the objects. For example, if the object is a person, the weight, shape, size, and carried possessions can be described. This description of the identified object and the identity of the identified object are examples of identification data generated by the identification system 20. This identification data is sent to the monitoring unit 10 and processed as described below.

Typically, the design of the identification system 20 and of its identification sensors determines the type/quality/accuracy of identification and description obtained on the identified objects. For example, if the object is a person, face recognition using vision sensors provides a detailed description but lower identification accuracy compared to the RFID technology. Yet, both face recognition and RFID technology do not provide the information about the citizenship of the person that could be obtained with a passport check at a security point. Thus, the identification system 20 can be a combination of multi-sensor, multi-technology, and human-assisted systems of identification, making possible to reach the level of accuracy and description required by the application utilizing the monitoring system 100.

Referring to FIG. 1, the monitoring unit 10 receives the identification data from the identification system 20 and the tracking data from the tracking system 30. Moreover, the monitoring unit 10 merges and stores the identification data and the tracking data of each monitored object to form monitoring data for each monitored object. Before the identification data and the tracking data for each monitored object is merged, the monitoring system 100 interprets this data as representing tracking data for an object assigned the unique tracking identifier X and as identification data for an object identified as a person named John Smith. After the identification data and the tracking data for each monitored object is merged, the monitoring system 100 interprets this data as representing tracking data and identification data for a person named John Smith. In effect, the identification made by the identification system 20 replaces the unique tracking identifier assigned by the tracking system 30 from the perspective of the monitoring system 100.

US 7,057,509 B2

5

Additionally, the monitoring unit 10 is configured to process the monitoring data (merged tracking data and identification data for each monitored object). Hence, the monitoring unit 10 can analyze and query the monitoring data, as needed.

FIG. 2 illustrates a block diagram of the monitoring unit 10 of FIG. 1 in accordance with an embodiment of the present invention. As depicted in FIG. 2, the monitoring unit 10 has an identification system interface 210 for receiving identification data from the identification system 20 (FIG. 1). Additionally, the monitoring unit 10 has a tracking system interface 220 for receiving tracking data from the tracking system 30 (FIG. 1). Furthermore, the monitoring unit 10 includes a merging unit 230 for merging and storing the identification data and the tracking data of each monitored object to form monitoring data 235 for each monitored object. The merging unit 230 includes the monitoring data 235. Also, the monitoring unit 10 has an analyzer unit 240 for processing the monitoring data 235. The components of the monitoring unit 10 can be implemented in hardware, software, or a combination of software and hardware. It should be understood that the monitoring unit 10 can be implemented differently than that shown in FIG. 2.

The merging unit is coupled to the identification system interface 210, the tracking system interface 220, and the analyzer unit 240 via connections 215, 225, and 242, respectively. Moreover, the analyzer unit 240 is configured to generate messages via line 245, as will be described below.

FIG. 3 illustrates data structure of the monitoring data 235 in accordance with an embodiment of the present invention. In an embodiment, a computer-readable medium has stored therein the data structure of the monitoring data 235. Examples of a computer-readable medium include a magnetic disk, CD-ROM, an optical medium, a floppy disk, a flexible disk, a hard disk, a magnetic tape, a RAM, a ROM, a PROM, an EPROM, a flash-EPROM, or any other medium from which a computer can read.

As shown in FIG. 3, the data structure of the monitoring data 235 includes a plurality of monitoring data groups 310, 320, and 330. Each monitoring data group 310–330 is associated with one of a plurality of monitored objects 342, 344, and 346. Moreover, each monitoring data group 310–330 includes identification data 350A–350C associated with a monitored object 342, 344, and 346. The identification data 350A–350C is generated when the monitored object 342, 344, and 346 is located at an identification gateway (as will be described below in connection with FIGS. 4 and 5) of a monitored environment. The identification system 20 (FIG. 1) generates the identification data 350A–350C. Each monitoring data group 310–330 includes tracking data 360A–360C associated with the monitored object 342, 344, and 346. The tracking data 360A–360C is generated when the monitored object 342, 344, and 346 is located within the monitored environment. The tracking system 30 (FIG. 1) generates the tracking data 360A–360C.

The identification gateway (e.g., 520A and 520B of FIG. 5) represents a location where the monitored object 342, 344 and 346 interfaces with the identification system 20 (FIG. 1). As described above, the identification system 20 can utilize automated identification systems, human-assisted identification systems, or a combination of both. As an example, a police officer that checks the passport number at a security checkpoint of a monitored person can represent an identification gateway. As another example, location of identification sensors based on pattern recognition (e.g., face recognition, iris recognition, fingerprint recognition, voice

6

recognition, etc.) or based on RFID (radio frequency identification) technology can also represent an identification gateway.

Moreover, since the monitored environment (e.g., 510 and 540 of FIG. 5) is (physically or logically) partitioned from a non-monitored environment, the identification gateways (or checkpoints) 520A and 520B can be utilized to determine whether monitored objects have entered or left the monitored environment 510 and 540. The monitored objects transition from the non-monitored environment to the monitored environment and vice versa via an identification gateway (e.g., 520A). Moreover, monitored objects transition from the monitored environment 510 to another monitored environment 540 and vice versa via an identification gateway (e.g., gateway 520B).

As described above, description of the identified object and the identity of the identified object are examples of identification data 350A–350C received from the identification system 20. Similarly, the description of the tracked object, the trajectory of the tracked object, and the unique tracking identifier are examples of tracking data 360A–360C received from the tracking system 30. In the monitoring data 235, the identified object and the tracked object are merged into the monitored object 342, 344, and 346.

As will be described below, the monitoring data group 310–330 of each monitored object 342, 344, and 346 is enriched with information provided when the identification gateways (will be described below in connection with FIGS. 4 and 5) identify the monitored objects 342, 344, and 346. In an embodiment, new descriptive information is generated at the identification gateways each time the monitored object 342, 344, and 346 is identified at any identification gateway and becomes part of the identification data 350A–350C.

Moreover, the monitoring data group 310–330 of each monitored object 342, 344, and 346 is enriched when the tracking system 30 (FIG. 1) provides tracking data 360A–360C having trajectories of the monitored objects 342, 344, and 346, enabling determination of interactions between monitored objects 342, 344, and 346 by comparing the trajectories.

FIG. 4 illustrates a flow chart showing a method 400 of monitoring an object in accordance with an embodiment of the present invention. Reference is made to FIGS. 1–3 and 5. In an embodiment, the method 400 is configured as computer-executable instructions stored in a computer-readable medium, such as a magnetic disk, CD-ROM, an optical medium, a floppy disk, a flexible disk, a hard disk, a magnetic tape, a RAM, a ROM, a PROM, an EPROM, a flash-EPROM, or any other medium from which a computer can read. FIG. 5 illustrates operation of identification system 20 and tracking system 30 in accordance with an embodiment of the present invention. As depicted in FIG. 5, the identification system 20 and the tracking system 30 are deployed to monitor an object(s) in a monitored environment 510. In particular, the monitored environment 510 is (physically or logically) partitioned from a non-monitored environment 530. The monitored environment 510 has one or more identification gateways (or checkpoints) 520A and 520B. In an embodiment, monitored objects transition from the non-monitored environment 530 to the monitored environment 510 and vice versa via an identification gateway (e.g., gateway 520A). Moreover, monitored objects transition from the monitored environment 510 to another monitored environment 540 and vice versa via an identification gateway (e.g., gateway 520B). The monitored environment 510 can have a wide range of sizes and shapes. Examples of monitored environments include a floor of a building (e.g.,

US 7,057,509 B2

7

an airport, a warehouse, a data center, etc.), a building, a room, a portion of a room, an aisle, etc.

As shown in FIG. 5, the tracking system 30 tracks objects in each monitored environment 510 and 540. The tracking system 30 can be a single system or a collection of subsystems functionally integrated or functionally independent of one another. This flexibility allows implementation of different technologies for providing tracking functionality within the same monitored environment or in different monitored environments. For example, it is possible to implement infrared cameras in some monitored environments (e.g., where the face detection is essential) and to implement video cameras in other monitored environments (e.g., where moving carts should be tracked).

Continuing with FIG. 5, the identification system 20 is deployed such that when the monitored object is located at and transitions through any identification gateway 520A and 520B, the monitored object is identified by the identification system 20. In an embodiment, the identification system 20 can be implemented as several identification subsystems that are functionally integrated or functionally independent of each other. This flexibility allows implementation of different technologies for providing identification functionality within the same identification gateway or in different identification gateways.

Referring again to FIG. 4, at Step 410, identification data is generated, where the identification data is associated with each monitored object located at an identification gateway (e.g., gateway 520A or gateway 520B) of a monitored environment. The identification data is generated by an identification system 20 deployed such that when the monitored object is located at and transitions through any identification gateway 520A and 520B, the monitored object is identified by the identification system 20.

Continuing with Step 420, tracking data is generating, where the tracking data is associated with each monitored object located within the monitored environment 510 and 540. The tracking data is generated by the tracking system 30.

At Step 430, the identification data and the tracking data of each monitored object is merged and stored, forming the monitoring data for each monitored object. The monitoring data can be processed, as needed.

FIG. 6 illustrates monitored environments in accordance with an embodiment of the present invention. As shown in FIG. 6, the tracking system (not shown) and the identification system (not shown) of the monitoring system 100 of FIG. 1 have been deployed to monitor objects in three separate monitored environments 610, 620, and 630. Moreover, the identification gateways 650A–650E are also depicted.

Operation of the monitoring system of FIG. 1 in accordance with an embodiment of the present invention is illustrated in FIG. 7. The tracking system 30 tracks the monitored object 770 within the monitored environment 710. Moreover, the identification system 20 is deployed such that when the monitored object 770 is located at and transitions through any identification gateway 720A and 720B, the monitored object 770 is identified by the identification system 20.

At $t=T1$, the object 770 is located at identification gateway 720A. Thus, the identification system 20 identifies the object 770. Also, at $t=T1$, the tracking system 30 detects the object 770 and starts tracking the object 770. The monitoring unit 10 receives the identification data (e.g., the name John Smith, the passport number X, identification occurred at identification gateway 720A at $t=T1$, etc.) from the identification system 20.

8

Similarly, the monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code Y is positioned at P1 at $t=T1$, where P1 represents the identification gateway 720A) from the tracking system 30. Since the identification data and the tracking data indicate object 770 is at identification gateway 720A (or position P1) at $t=T1$, the monitoring unit 10 determines that the received identification data and the tracking data should be merged since the position (as provided by the identification system 20 and the tracking system 30) is the same within the same time (as provided by the identification system 20 and the tracking system 30). Hence, the monitoring unit 10 merges the received identification data and the tracking data to form the monitoring data for monitored object 770. Moreover, if the monitored object 770 already is associated with a monitoring data group (as described in FIG. 3), the merged data is stored with the monitoring data group. Typically, the identification data enables the monitoring unit 10 to determine whether the monitored object 770 is associated with a monitoring data group. If the monitored object 770 is not associated with a monitoring data group, a monitoring data group is created for the monitored object 770 and the merged data is stored with the created monitoring data group.

At $t=T2$, the tracking system 30 continues to track the monitored object 770. The monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code Y is positioned at P2 at $t=T2$) from the tracking system 30. The monitoring unit 10 determines whether the unique tracking code Y has been assigned to any monitored object. Since monitored object 770 was assigned the unique tracking code Y by the tracking system 30, the monitoring unit 10 stores the tracking data with the monitoring data for monitored object 770. Since the monitored object 770 already is associated with a monitoring data group (as described in FIG. 3), the tracking data is stored with the monitoring data group. Moreover, the tracking data in the monitoring data group would indicate the monitored object 770 has the trajectory 780 in the monitored environment 710.

At $t=T3$, the monitored object 770 is located at identification gateway 720B. Thus, the identification system 20 identifies the monitored object 770. Moreover, the tracking system 30 continues to track the monitored object 770. The monitoring unit 10 receives the identification data (e.g., the name John Smith, the passport number X, identification occurred at identification gateway 720B at $t=T3$, etc.) from the identification system 20. Similarly, the monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code Y is positioned at P3 at $t=T3$, where P3 represents the identification gateway 720B) from the tracking system 30. Since the identification data and the tracking data indicate monitored object 770 is at identification gateway 720B (or position P3) at $t=T3$, the monitoring unit 10 determines that the received identification data and the tracking data should be merged since the position (as provided by the identification system 20 and the tracking system 30) is the same within the same time (as provided by the identification system 20 and the tracking system 30). Hence, the monitoring unit 10 merges the received identification data and the tracking data and stores it with the monitoring data of the monitored object 770. Since the monitored object 770 already is associated with a monitoring data group (as described in FIG. 3), the merged data is stored with the monitoring data group. Moreover, the tracking data in the monitoring data group would indicate the monitored object 770 has the trajectory 790 in the monitored environment 710.

US 7,057,509 B2

9

If the identification gateway 720B provides a transition from the monitored environment 710 to a non-monitored environment, the monitored object 770 is no longer monitored by the monitoring system 100. However, if the identification gateway 720B provides a transition from the monitored environment 710 to another monitored environment, the tracking system 30 would continue to track the monitored object 770 in the other monitored environment.

The monitoring data 235 (FIGS. 2 and 3) can be analyzed to find the cause of a problem in a particular area within the monitored environment 710. For example, if the monitored environment 710 is an aisle in a warehouse or data center, an analysis of the monitoring data 235 can identify the monitored objects that were in the particular area within the monitored environment 710.

In addition to supporting identification functionality, each identification gateway provides the opportunity to compare the content of the monitoring data of a monitored object at different times. Additionally, each identification gateway provides the opportunity to recover from errors arising from the tracking system 30.

FIG. 8 illustrates comparison functionality of the monitoring system 100 of FIG. 1 in accordance with an embodiment of the present invention. The tracking system 30 tracks the monitored object 37 within the monitored environment 810. Moreover, the identification system 20 is deployed such that when the monitored object 37 is located at and transitions through any identification gateway 820A and 820B, the monitored object 37 is identified by the identification system 20.

At $t=T1$, the monitored object 37 is located at identification gateway 820A (or Gateway1). Thus, the identification system 20 identifies the monitored object 37. Moreover, the tracking system 30 detects the monitored object 37 and starts tracking the monitored object 37. The monitoring unit 10 receives the identification data (e.g., the name John Smith, the passport number c383902, identification occurred at identification gateway 820A at $t=T1$, a snapshot, number of suitcases is 2, etc.) from the identification system 20. Similarly, the monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code Z is positioned at P1 at $t=T1$, where P1 represents the identification gateway 820A) from the tracking system 30. Since the identification data and the tracking data indicate monitored object 37 is at identification gateway 820A (or position P1) at $t=T1$, the monitoring unit 10 merges the received identification data and the tracking data and stores it with the monitoring data of the monitored object 37. Since the monitored object 37 already is associated with a monitoring data group (as described in FIG. 3), the merged data is stored with the monitoring data group.

At $t=T2$, the monitored object 37 is located at identification gateway 820B (or Gateway2). Thus, the identification system 20 identifies the monitored object 37. Moreover, the tracking system 30 continues to track the monitored object 770. The monitoring unit 10 receives the identification data (e.g., the name John Smith, the passport number c383902, identification occurred at identification gateway 820B at $t=T1$, a snapshot, the number of suitcases is 1, etc.) from the identification system 20. Similarly, the monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code Z is positioned at P2 at $t=T2$, where P2 represents the identification gateway 820B) from the tracking system 30. Since the identification data and the tracking data indicate monitored object 37 is at identification gateway 820B (or position P2) at $t=T2$, the monitoring unit 10 merges the received identification data and the tracking data and

10

stores it with the monitoring data of the monitored object 37. Since the monitored object 37 already is associated with a monitoring data group (as described in FIG. 3), the merged data is stored with the monitoring data group. Moreover, the tracking data in the monitoring data group would indicate the monitored object 37 has the trajectory 890 in the monitored environment 810.

Since the monitored object 37 is identified at identification gateway 820B (or Gateway2) and identification data is generated, the monitoring unit 10 is able to compare the new content of the monitoring data (generated at identification gateway 820B) with the prior content of the monitoring data (generated at identification gateway 820A). In an embodiment, the analyzer unit 240 (FIG. 2) provides this functionality. The result of the comparison can lead to the generation of a warning message. For example, if there is a mismatch, a warning message 870 can be generated by the analyzer unit 240 via line 245 (FIG. 2). Under the facts of FIG. 8, "number of suitcases is changed" warning message 870 would be generated because the monitored object 37 had two suitcases at identification gateway 820A at $t=T1$ but had only one suitcase at identification gateway 820B at $t=T2$.

As discussed above, each identification gateway provides the opportunity to recover from errors arising from the tracking system 30. One type of error arising from the tracking system 30 is caused by losing track of a monitored object within the monitored environment. The identification data generated by identifying the monitored object at the identification gateway after losing track of the monitored object facilitates recovering from this type of error. This case will be illustrated in FIG. 9. Another type of error arising from the tracking system 30 is caused by interaction between monitored objects, reducing the tracking system's 30 level of certainty related to correct association between tracking data and the monitored objects. For example, two monitored objects may move to a location where they are near each other such that the tracking system 30 becomes confused and is unable to distinguish the two monitored objects. Then, the two objects separate. After the separation, the tracking system 30 will detect two monitored objects but will be unable to associate tracking data with the correct monitored object. The identification data generated by identifying a monitored object at the identification gateway after the tracking system 30 reduced the level of certainty facilitates recovering from this type of error. This case will be illustrated in FIGS. 10A and 10B.

FIG. 9 illustrates a first error recovery functionality of the monitoring system 100 of FIG. 1 in accordance with an embodiment of the present invention. The tracking system 30 tracks the monitored object 37 within the monitored environment 910. Moreover, the identification system 20 is deployed such that when the monitored object 37 is located at and transitions through any identification gateway 920A and 920B, the monitored object 37 is identified by the identification system 20.

At $t=T1$, the monitored object 37 is located at identification gateway 920A. Thus, the identification system 20 identifies the monitored object 37. Moreover, the tracking system 30 detects the monitored object 37 and starts tracking the monitored object 37. The monitoring unit 10 receives the identification data (e.g., the name John Smith, identification occurred at identification gateway 920A at $t=T1$, etc.) from the identification system 20. Similarly, the monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code Z is positioned at P1 at $t=T1$, where P1 represents the identification gateway 920A) from the tracking system 30. Since the identification data and the tracking

US 7,057,509 B2

11

data indicate monitored object 37 is at identification gateway 920A (or position P1) at $t=T1$, the monitoring unit 10 determines that the received identification data and the tracking data should be merged since the position (as provided by the identification system 20 and the tracking system 30) is the same within the same time (as provided by the identification system 20 and the tracking system 30). Hence, the monitoring unit 10 merges the received identification data and the tracking data and stores it with the monitoring data of the monitored object 37. Since the monitored object 37 already is associated with a monitoring data group (as described in FIG. 3), the merged data is stored with the monitoring data group.

At $t=T2$, the tracking system 30 loses track of the monitored object 37. However, the tracking data in the monitoring data group would indicate the monitored object 37 has the trajectory 994 in the monitored environment 910 before loss of tracking. Later, at $t=T3$, the tracking system 30 detects an object 992. From the perspective of the tracking system 30, the assumption can be made that object 992 is monitored object 37 after considering time and position. However, if this assumption is incorrect, the monitoring unit 10 will incorrectly merge tracking data and identification data, raising the possibility that the ability to unwind the incorrectly merged data may be lost.

Thus, the tracking system 30 starts tracking the object 992, after assigning it the unique tracking code M. The monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code M is positioned at P3 at $t=T3$, etc.) from the tracking system 30. However, the monitoring unit 10 will determine that in the monitoring data 235 (FIGS. 2 and 3) none of the previously monitored objects has been assigned the unique tracking code M by the tracking system 30. Thus, the monitoring unit creates a monitoring data group for the monitored object 992 and stores the tracking data in the created monitoring data group. Furthermore, at $t=T4$, the tracking system 30 continues to track the monitored object 992. The monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code M is positioned at P4 at $t=T4$, etc.) from the tracking system 30. Since the monitored object 992 already is associated with a monitoring data group (as described in FIG. 3), the tracking data is stored with the monitoring data group. Moreover, the tracking data in the monitoring data group would indicate the monitored object 992 has the trajectory 995 in the monitored environment 910.

At $t=T5$, the monitored object 992 is located at identification gateway 920B. Thus, the identification system 20 identifies the monitored object 992 as being the monitored object 37. Moreover, the tracking system 30 continues to track the monitored object 992. The monitoring unit 10 receives the identification data (e.g., the name John Smith, identification occurred at identification gateway 920B at $t=T5$, etc.) from the identification system 20. Similarly, the monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code M is positioned at P5 at $t=T5$, where P5 represents the identification gateway 920B) from the tracking system 30. Since the identification data indicates that monitored object 992 is monitored object 37 and the tracking data indicates monitored object 992 is at identification gateway 920B (or position P5) at $t=T5$, the monitoring unit 10 merges the received identification data and the tracking data and stores it with the monitoring data of the monitored object 37. Since the monitored object 37 already is associated with a monitoring data group (as described in FIG. 3), the merged data is stored with the monitoring data group.

12

Additionally, the monitoring unit 10 modifies the monitoring data 235 (FIGS. 2 and 3) to merge the monitoring data group associated with monitored object 992 (which was assigned the tracking code M by the tracking system 30) and the monitoring data group associated with monitored object 37 (which was assigned the tracking code Z by the tracking system 30). Moreover, the tracking data in the monitoring data group would indicate the monitored object 37 has the trajectory 990 in the monitored environment 910. In an embodiment, the analyzer unit 240 (FIG. 2) provides this error recovery functionality that enables recovery from an error arising from the tracking system 30 caused by losing track of a monitored object within the monitored environment.

FIGS. 10A and 10B illustrate a second error recovery functionality of the monitoring system 100 of FIG. 1 in accordance with an embodiment of the present invention. As depicted in FIGS. 10A and 10B, the tracking system 30 tracks the monitored objects 37 and 5 within the monitored environment 1010. Moreover, the identification system 20 is deployed such that when the monitored object 37 or the monitored object 5 is located at and transitions through any identification gateway 1020A and 1020B, the monitored object 37 or 5 is identified by the identification system 20.

Referring to FIG. 10A, at $t=T1$, the tracking system 30 continues to track the monitored objects 37 and 5, which have previously transitioned through any identification gateway 1020A and 1020B for identification by the identification system 20. The tracking system 30 has assigned unique tracking code X to monitored object 37 and has assigned unique tracking code W to monitored object 5. Moreover, the tracking system 30 is 100% certain that it is associating the tracking data with the correct monitored object. The monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code X is positioned at P1A at $t=T1$, object assigned unique tracking code W is positioned at P1B at $t=T1$, etc.) from the tracking system 30. Since the monitored objects 37 and 5 already are associated with a corresponding monitoring data group (as described in FIG. 3), the tracking data for each monitored object 37 and 5 is stored with the corresponding monitoring data group.

At $t=T2$, the tracking system 30 continues to track the monitored objects 37 and 5. However, the monitored objects 37 and 5 move to a location where they are near each other such that the tracking system 30 becomes confused and is unable to distinguish the two monitored objects 37 and 5. Although the tracking system 30 is unable to distinguish the two monitored objects 37 and 5, the tracking system 30 determines that they are at positioned at P2 at $t=T2$. The monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code X is positioned at P2 at $t=T2$, object assigned unique tracking code W is positioned at P2 at $t=T2$, etc.) from the tracking system 30. Since the monitored objects 37 and 5 already are associated with a corresponding monitoring data group (as described in FIG. 3), the tracking data for each monitored object 37 and 5 is stored with the corresponding monitoring data group. Moreover, the tracking data in the corresponding monitoring data groups would indicate the monitored object 37 has the trajectory 1091 in the monitored environment 1010 while the monitored object 5 has the trajectory 1092 in the monitored environment 1010.

Continuing with FIG. 10A, at $t=T3$, the two monitored objects 37 and 5 have separated. After the separation, the tracking system 30 detects two monitored objects 1001 and 1002 but is unable to associate tracking data with the correct monitored object (e.g., monitored object 37 or monitored

US 7,057,509 B2

13

object 5) with 100% certainty. In fact, the tracking system is 50% certain that monitored object 1001 is monitored object 37 and is 50% certain that monitored object 1001 is monitored object 5. Similarly, the tracking system is 50% certain that monitored object 1002 is monitored object 37 and is 50% certain that monitored object 1002 is monitored object 5. Thus, the tracking system 30 has reduced the level of certainty related to correction association between tracking data and monitored objects. The monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code X or unique tracking code W is positioned at P3A at t=T3, object assigned unique tracking code W or unique tracking code X is positioned at P3B at t=T3, etc.) from the tracking system 30. Since the level of certainty related to correct association between tracking data and monitored objects is not 100%, the monitoring unit 10 stores this tracking data separately from the monitoring data groups associated with monitored object 37 and monitored object 5.

At t=T4, the tracking system 30 continues to track monitored objects 1001 and 1002. As discussed above, the tracking system is 50% certain that monitored object 1001 is monitored object 37 and is 50% certain that monitored object 1001 is monitored object 5. Similarly, the tracking system is 50% certain that monitored object 1002 is monitored object 37 and is 50% certain that monitored object 1002 is monitored object 5. The monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code X or unique tracking code W is positioned at P4A at t=T4, object assigned unique tracking code W or unique tracking code X is positioned at P4B at t=T4, etc.) from the tracking system 30. Since the level of certainty related to correct association between tracking data and monitored objects is not 100%, the monitoring unit 10 continues to store this tracking data separately from the monitoring data groups associated with monitored object 37 and monitored object 5. Moreover, the tracking data in the monitoring data group would indicate the monitored object 1001 has the trajectory 1093 in the monitored environment 1010 while the monitored object 1002 has the trajectory 1094 in the monitored environment 1010.

As depicted in FIG. 10A, at t=T5, the monitored object 1002 is located at identification gateway 1020B. Thus, the identification system 20 identifies the monitored object 1002 as being the monitored object 37. Moreover, the tracking system 30 continues to track the monitored object 37 and 1001. The monitoring unit 10 receives the identification data (e.g., the name John Smith, identification occurred at identification gateway 1020B at t=T5, etc.) from the identification system 20. Similarly, the monitoring unit 10 receives the tracking data (e.g., object assigned unique tracking code X or unique tracking code W is positioned at P5A at t=T5 where P5A represents the identification gateway 1020B, object assigned unique tracking code W or unique tracking code X is positioned at P5B at t=T5, etc.) from the tracking system 30. Since the identification data indicates that monitored object 1002 is monitored object 37 and the tracking data indicates monitored object 1002 is at identification gateway 1020B (or position P5A) at t=T5, the monitoring unit 10 merges the received identification data associated with monitored object 1002 and the tracking data associated with monitored object 1002 and stores it with the monitoring data of the monitored object 37. Since the monitored object 37 already is associated with a monitoring data group (as described in FIG. 3), the merged data is stored with the corresponding monitoring data group.

Referring to FIG. 10B, since the monitoring unit 10 is 100% certain that monitored object 1002 is monitored object

14

37 due to the identification system 20, the monitoring unit 10 can determine with 100% certainty that monitored object 1001 is monitored object 5. Thus, the monitoring unit 10 modifies the monitoring data 235 (FIGS. 2 and 3) to merge the monitoring data group associated with monitored object 1001 and the monitoring data group associated with monitored object 5. Furthermore, the monitoring unit 10 modifies the monitoring data 235 (FIGS. 2 and 3) to merge the monitoring data group associated with monitored object 1001 and the monitoring data group associated with monitored object 5. Therefore, the tracking data in the monitoring data groups would indicate the monitored object 37 has the trajectory 1099 in the monitored environment 1010 while the monitored object 5 has the trajectory 1098 in the monitored environment 1010. In an embodiment, the analyzer unit 240 (FIG. 2) provides this error recovery functionality that enables recovery from an error arising from the tracking system 30 caused by confusion and inability to distinguish monitored objects that are near each other.

In an embodiment, the invention is configured as computer-executable instructions stored in a computer-readable medium, such as a magnetic disk, CD-ROM, an optical medium, a floppy disk, a flexible disk, a hard disk, a magnetic tape, a RAM, a ROM, a PROM, an EPROM, a flash-EPROM, or any other medium from which a computer can read.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto and their equivalents.

What is claimed is:

1. A monitoring apparatus comprising:
 - a first interface for receiving identification data from an identification system;
 - a second interface for receiving tracking data from a tracking system;
 - a merging unit for merging and storing said identification data and said tracking data of each monitored object to form monitoring data for each monitored object; and
 - an analyzer unit for processing said monitoring data, wherein said analyzer unit facilitates comparing new content of said monitoring data of a monitored object with prior content of said monitoring data of said monitored object.
2. The monitoring apparatus as recited in claim 1 wherein said analyzer unit generates a message based on a result of said comparison of said new and said prior content.
3. The monitoring apparatus as recited in claim 1 wherein if said tracking system loses track of a first monitored object within a monitored environment, said analyzer unit determines whether a second monitored object represents said first monitored object lost by said tracking system.
4. The monitoring apparatus as recited in claim 3 wherein said analyzer unit makes said determination in response to said identification system identifying said first monitored object after said tracking system lost track of said first monitored object, and wherein said analyzer unit modifies said monitoring data based on said determination.

US 7,057,509 B2

15

5. The monitoring apparatus as recited in claim 1 wherein if said tracking system tracks a plurality of monitored objects within a monitored environment and if said tracking system reduces certainty related to correct association between tracking data and said monitored objects, said analyzer unit increases said certainty in response to said identification system identifying at least one of said monitored objects after said tracking system reduced said certainty.

6. The monitoring apparatus as recited in claim 5 wherein said analyzer unit modifies said monitoring data upon increasing said certainty.

7. A monitoring system comprising:

an identification system for generating identification data associated with each monitored object located at an identification gateway of a monitored environment;

a tracking system for generating tracking data associated with each monitored object located within said monitored environment; and

a monitoring unit coupled to said identification and tracking systems, wherein said monitoring unit merges and stores said identification data and said tracking data of each monitored object to form monitoring data for each monitored object, wherein said monitoring unit compares new content of said monitoring data of a monitored object with prior content of said monitoring data of said monitored object.

8. The monitoring system as recited in claim 7 wherein said monitoring unit processes said monitoring data.

9. The monitoring system as recited in claim 7 wherein said monitoring unit generates a message based on a result of said comparison of said new and said prior content.

10. The monitoring system as recited in claim 8 wherein if said tracking system loses track of a first monitored object within said monitored environment, said monitoring unit determines whether a second monitored object represents said first monitored object lost by said tracking system.

11. The monitoring system as recited in claim 10 wherein said monitoring unit makes said determination in response to said identification system identifying said first monitored object after said tracking system lost track of said first monitored object, and wherein said monitoring unit modifies said monitoring data based on said determination.

12. The monitoring system as recited in claim 8 wherein if said tracking system tracks a plurality of monitored objects within said monitored environment and if said tracking system reduces certainty related to correct association between tracking data and said monitored objects, said monitoring unit increases said certainty in response to said identification system identifying at least one of said monitored objects after said tracking system reduced said certainty.

13. The monitoring system as recited in claim 12 wherein said monitoring unit modifies said monitoring data upon increasing said certainty.

14. A computer-readable medium having stored therein a data structure comprising:

A plurality of monitoring data groups, each monitoring data group associated with one of a plurality of monitored objects, wherein each monitoring data group includes:

identification data associated with a monitored object, wherein said identification data is generated when said monitored object is located at an identification gateway of a monitored environment;

16

tracking data associated with said monitored object, wherein said tracking data is generated when said monitored object is located within said monitored environment; and

merged data that is based on the comparison of new content of monitoring data of a monitored object and prior content of said monitoring data of said monitored object.

15. The computer-readable medium as recited in claim 14 wherein said identification data is generated by an identification system.

16. The computer-readable medium as recited in claim 14 wherein said tracking data is generated by a tracking system.

17. A method of monitoring an object, said method comprising:

generating identification data associated with each monitored object located at an identification gateway of a monitored environment, wherein said generating identification data is performed by an identification system; generating tracking data associated with each monitored object located within said monitored environment, wherein said generating tracking data is performed by a tracking system;

merging and storing said identification data and said tracking data of each monitored object to form monitoring data for each monitored object; and comparing new content of said monitoring data of a monitored object with prior content of said monitoring data of said monitored object.

18. The method as recited in claim 17 further comprising: processing said monitoring data.

19. The method as recited in claim 18 wherein said processing further includes:

generating a message based on a result of said comparison of said new and said prior content.

20. The method as recited in claim 18 wherein if said tracking system loses track of a first monitored object within said monitored environment, said processing includes:

determining whether a second monitored object represents said first monitored object lost by said tracking system.

21. The method as recited in claim 20 wherein said determining is performed in response to said identification system identifying said first monitored object after said tracking system lost track of said first monitored object, and wherein said processing further includes modifying said monitoring data based on said determining step.

22. The method as recited in claim 18 wherein if said tracking system tracks a plurality of monitored objects within said monitored environment and if said tracking system reduces certainty related to correct association between tracking data and said monitored objects, said processing includes:

increasing said certainty in response to said identification system identifying at least one of said monitored objects after said tracking system reduced said certainty.

23. The method as recited in claim 22 wherein said processing further includes modifying said monitoring data upon increasing said certainty.

24. A computer-readable medium comprising computer-executable instructions stored therein for performing a method of monitoring an object, said method comprising:

generating identification data associated with each monitored object located at an identification gateway of a monitored environment, wherein said generating identification data is performed by an identification system;

US 7,057,509 B2

17

generating tracking data associated with each monitored object located within said monitored environment, wherein said generating tracking data is performed by a tracking system;

merging and storing said identification data and said tracking data of each monitored object to form monitoring data for each monitored object; and

comparing new content of said monitoring data of a monitored object with prior content of said monitoring data of said monitored object.

25. The computer-readable medium as recited in claim 24 wherein said method further comprises:

processing said monitoring data.

26. The computer-readable medium as recited in claim 25 wherein said processing further includes:

generating a message based on a result of said comparison of said new and said prior content.

27. The computer-readable medium as recited in claim 25 wherein if said tracking system loses track of a first monitored object within said monitored environment, said processing includes:

determining whether a second monitored object represents said first monitored object lost by said tracking system.

18

28. The computer-readable medium as recited in claim 27 wherein said determining is performed in response to said identification system identifying said first monitored object after said tracking system lost track of said first monitored object, and wherein said processing further includes modifying said monitoring data based on said determining step.

29. The computer-readable medium as recited in claim 25 wherein if said tracking system tracks a plurality of monitored objects within said monitored environment and if said tracking system reduces certainty related to correct association between tracking data and said monitored objects, said processing includes:

increasing said certainty in response to said identification system identifying at least one of said monitored objects after said tracking system reduced said certainty.

30. The computer-readable medium as recited in claim 29 wherein said processing further includes modifying said monitoring data upon increasing said certainty.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 7,057,509 B2
APPLICATION NO. : 10/881975
DATED : June 6, 2006
INVENTOR(S) : Giovanni Gualdi et al.

Page 1 of 1

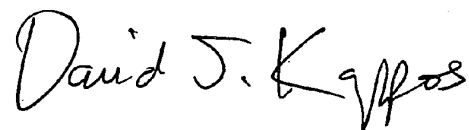
It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 15, line 59, in Claim 14, delete "compnsing:" and insert -- comprising: --, therefor.

In column 16, line 57, in Claim 22, delete "tacking" and insert -- tracking --, therefor.

Signed and Sealed this

Nineteenth Day of January, 2010

A handwritten signature in black ink, reading "David J. Kappos". The signature is written in a cursive, flowing style with a large, stylized 'D' and 'K'.

David J. Kappos
Director of the United States Patent and Trademark Office