

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS, INC.,
Petitioner

IPR2023-00663
U.S. Patent No. 9,998,607

**PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104**

TABLE OF CONTENTS

Petitioner’s Exhibit List	5
I. Introduction.....	7
II. Grounds for standing	7
III. Note.....	8
IV. Technology background	8
V. Summary of the ’607 patent	8
VI. Prosecution history	10
VII. Effective priority date of the ’607 patent	11
VIII. Level of ordinary skill in the art	11
IX. Claim construction.....	12
X. Relief requested and the reasons for the requested relief.....	13
XI. Identification of how the claims are unpatentable.....	13
A. Challenged claims	13
B. Statutory grounds for challenges.....	13
C. Ground 1: Claims 1-3 and 5-6 are obvious under 35 U.S.C. § 103 over de Castro and Robb	15
1. De Castro.....	15
2. Robb	18
3. Reasons to combine de Castro and Robb.....	21
4. Claim 1	26

5.	Claim 2	55
6.	Claim 3	59
7.	Claim 5	64
8.	Claim 6	66
XII.	PTAB Discretion Should Not Preclude Institution	67
A.	No basis for 35 U.S.C. § 325(d) denial	67
B.	The <i>Fintiv</i> Factors Favor Institution under § 314	68
1.	Whether evidence exists that a stay may be granted if a proceeding is instituted	68
2.	Proximity of the court’s trial date to the board’s projected statutory deadline for a final written decision	69
3.	Investment in the parallel proceeding	70
4.	Overlap between issues raised in the petition and in the parallel proceeding	70
5.	Whether the petitioner is a defendant	71
6.	Other circumstances that impact the Board’s exercise of discretion, including the merits	71
C.	Discretionary denial under <i>General Plastic</i> is not appropriate	72
XIII.	Conclusion	72
XIV.	Mandatory notices	73
A.	Real party-in-interest	73
B.	Related matters	73
C.	Lead and back-up counsel and service information	73

Certificate of Word Count	75
Certificate of Service	76

PETITIONER’S EXHIBIT LIST

Ex.1001	U.S. Pat. No. 9,998,607 to Hunt et al. (“’607 patent”)
Ex.1002	Prosecution History of U.S. Pat. No. 9,998,607
Ex.1003	Declaration of Henry Houh under 37 C.F.R. § 1.68
Ex.1004	<i>Curriculum Vitae</i> of Henry Houh
Ex.1005	U.S. Pat. Pub. No. 2012/0016932 to de Castro <i>et al.</i>
Ex.1006	U.S. Pat. No. 8,126,722 to Robb <i>et al.</i>
Ex.1007	Zapier, “API Reference – Developers,” Jan. 23, 2013 web archive of https://zapier.com/developer/reference/ , available at Wayback Machine on Oct. 7, 2022
Ex.1008	Zapier, “Integrate your Web Services,” Jan. 23, 2013 web archive of https://zapier.com/developer/reference/ , available at Wayback Machine on Oct. 7, 2022
Ex.1009	Ben Langfeld, Jose de Castro, “XEP-0327: Rayo,” XMPP Standards Foundation (<i>archived</i> Aug. 18, 2013)
Ex.1010	U.S. Pat. No. 7,593,980 to Marascio et al.
Ex.1011	U.S. Pat. No. 7,496,054 to Taylor
Ex.1012	U.S. Pat. No. 9,557,889 to Raleigh et al.
Ex.1013	Declaration of Jose de Castro under 37 C.F.R. § 1.68
Ex.1014	Affidavit of Authenticity for Ex.1007 and Ex.1008
Ex.1015	Exhibit A to Ex.1014
Ex.1016	Affidavit of Authenticity for Ex.1009
Ex.1017	U.S. District Courts – Combined Civil and Criminal Federal Court Management Statistics (Dec. 31, 2022), available at: https://www.uscourts.gov/statistics/table/na/federal-court-

	management-statistics/2022/12/31-3
Ex.1018	Docket Control Order, <i>Zilkr Cloud Technologies, LLC v. Cisco Systems, Inc.</i> , No. 2-22-cv-00166-JRG-RSP, Dkt. No. 31 (E.D. Tex., Oct. 13, 2022)
Ex.1019	Plaintiff’s Notice of Disclosures, <i>Zilkr Cloud Technologies, LLC v. Cisco Systems, Inc.</i> , No. 2-22-cv-00166-JRG-RSP, Dkt. No. 26 (E.D. Tex., Sep. 22, 2022)
Ex.1020	Original Complaint for Patent Infringement, <i>Zilkr Cloud Technologies, LLC v. Cisco Systems, Inc.</i> , No. 2-22-cv-00166-JRG-RSP, Dkt. No. 1 (E.D. Tex., May 20, 2022)
Ex.1021	Director Katherine K. Vidal, “Interim Procedure for Discretionary Denials in AIA Post-Grant Proceedings with Parallel District Court Litigation,” June 21, 2022
Ex.1022	U.S. Pat. No. 8,600,836 to Collins.
Ex.1023	U.S. Pat. Pub. No. 2007/0033194 to Srinivas <i>et al.</i>
Ex.1024	Microsoft Pres Computer Dictionary (3d ed. 1997)
Ex.1025	Newton’s Telecom Dictionary (18th ed. 2002)
Ex.1026	Copyright Record for Ex.1024
Ex.1027	Copyright Record for Ex.1025
Ex.1028	PACER Calendar Events Report for Judge Rodney Gilstrap, January 22, 2024.

I. INTRODUCTION

Pursuant to 35 U.S.C. §§ 311, 314(a), and 37 C.F.R. § 42.100, Cisco Systems, Inc. (“Petitioner”) respectfully requests that the Board review and cancel as unpatentable under 35 U.S.C. §103 claims 1-3 and 5-6 (hereinafter, the “Challenged Claims”) of U.S. 9,998,607 (the “’607 patent,” Ex.1001).

The ’607 patent is directed to providing multiple third-party telecommunications applications and services to add to existing telephone service. Ex.1001, 1:31-33, 8:50-54. Such additional applications and services include controlling phone calls over the internet, messaging, and web services. Ex.1001, 9:38-49. The ’607 patent describes integrating third-party application interfaces and a telephone service provider interface into a common interface and publishing information about the applications in a catalog for users. Ex.1001, 12:9-14, 58:25-59. However, as shown below and confirmed in the Declaration of Dr. Henry Houh (Ex.1003), these concepts were well known in the prior art before the ’607 patent’s earliest priority date. E.g., Ex.1005, Fig. 14, [0038]-[0041], [0138]-[0142]; Ex.1006, 34:16-27. The prior art presented in this Petition renders obvious the Challenged Claims.

II. GROUNDS FOR STANDING

Petitioner certifies that the ’607 patent is eligible for IPR and that Petitioner is not barred or estopped from requesting IPR challenging the patent claims. 37

C.F.R. § 42.104(a).

III. NOTE

Petitioner cites to exhibits' original page numbers. **Emphasis** in quoted material has been added. Claim terms are presented in *italics*.

IV. TECHNOLOGY BACKGROUND

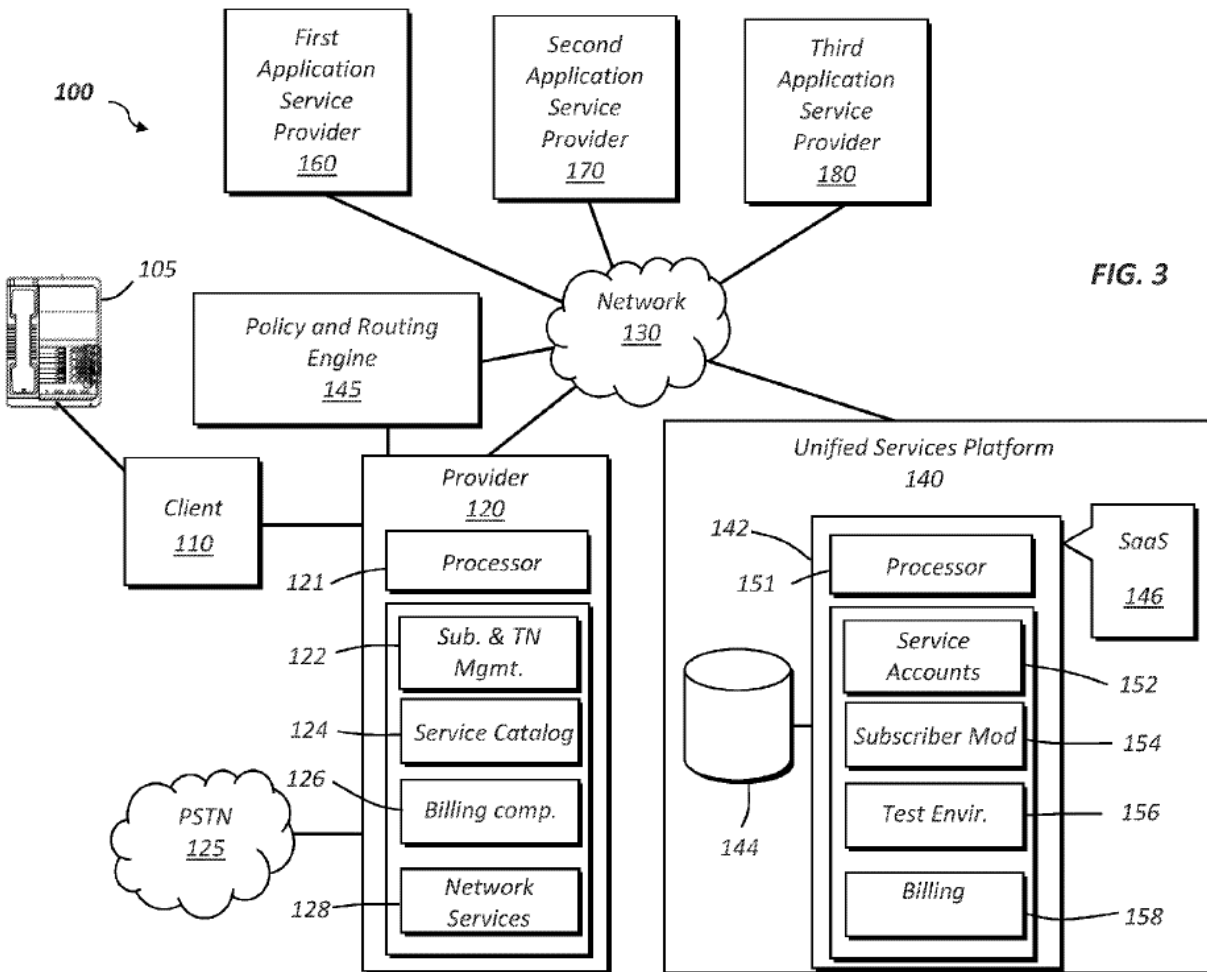
Dr. Houh's Declaration (Ex.1003) provides a technology background.
Ex.1003, ¶¶28-36.

V. SUMMARY OF THE '607 PATENT

According to the '607 patent, "the telephone number remains a valuable identifier for providing telephone services." Ex.1001, 1:29-30. Telephone service customers sometimes "wish to add services and applications to their existing telephone service." Ex.1001, 1:32-33. Examples of third-party telecommunications services and applications include session initiation protocol (SIP) for controlling phone call sessions over internet networks, and applications for messaging, collaboration, and web services. Ex.1001, 9:38-49. "However, many services and applications are supplied by third-party providers in different ecosystems," and as a result, they use different application programming interfaces (APIs) that may be incompatible with each other. Ex.1001, 1:32-33, 58:45-48. The '607 patent alleges that "new techniques for integrating advanced services and applications with legacy telephone service are desired." Ex.1001, 1:39-41; Ex.1003, ¶¶37-38.

The proposed solution, according to the '607 patent, is “integrating [a] plurality of API interfaces” to “provide services for use in association with a telephone number of a subscriber.” Ex.1001, 1:52-58. For instance, “two of the API interfaces are different, and at least one of the plurality of API interfaces from the plurality of application service provider systems is not compatible with the [network] provider API interface.” Ex.1001, Abstract, 1:59-63. To address such incompatibilities, the '607 patent merely describes “integrating [] each of the plurality of API interfaces with a unified services platform API interface,” which “provide[s] a common interface between the application service providers.” Ex.1001, 58:39-41, 13:35-42; Ex.1003, ¶¶39-40.

Figure 3 (below) illustrates unified services platform 140—which provides a common or unified API for the different APIs—connected via network 130 to client or device 110, telephone service provider 120, and application service providers 160, 170, 180. Ex.1001, Fig.3, 10:26-31, 10:51-52, 13:35-38; Ex.1003, ¶41.



Ex.1001, Fig.3

These and the other concepts recited in the Challenged Claims were known in the prior art before to the earliest priority date of the '607 patent. E.g., Ex.1005, [0038], [0119]; Ex.1006, 34:16-17, 34:25-27, 35:9-10; Ex.1003, ¶42.

VI. PROSECUTION HISTORY

The '607 patent's earliest priority claim is to provisional application 61/888,856, filed on October 9, 2013. Ex.1002, 350. The '607 patent was filed on April 8, 2016 as Appl. No. 15/094,433. Ex.1002, 467. The first Office action

rejected the claims as obvious over several references but found allowable subject matter in a dependent claim reciting:

...publishing, after integrating each of the plurality of API interfaces with the unified services platform, the definition of the service, the definition of the plan, and the one or more definitions of actions for each application service provider in a provisioning catalog.

Ex.1002, 67-71, 95.

Applicant amended the independent claims to incorporate this limitation.

Ex.1002, 45-46. The Patent Office then issued a notice of allowance on February 20, 2018, and the '607 patent issued on June 12, 2018. Ex.1002, 2, 18-19.

VII. EFFECTIVE PRIORITY DATE OF THE '607 PATENT

The '607 patent claims priority to a provisional application filed October 9, 2013, and all of the cited and relied-upon prior art predates this date. Accordingly, Petitioner has not undertaken a priority date analysis. Petitioner does not waive any right or opportunity it may have to dispute the priority date of the '607 patent in this or another forum where the issue is relevant.

VIII. LEVEL OF ORDINARY SKILL IN THE ART

A Person of Ordinary Skill in The Art ("POSITA") in October 2013 would have had a working knowledge of the telecommunications and networking art that is pertinent to the '607 patent. A POSITA would have had a bachelor's degree in electrical engineering, computer science, computer engineering, or an equivalent,

and two years of relevant experience in telecommunications and networking. Lack of professional experience can be remedied by additional education, and vice versa. Ex.1003, ¶22.

IX. CLAIM CONSTRUCTION

Claim terms in IPR are construed according to their “ordinary and customary meaning” to those of skill in the art. 37 C.F.R. § 42.100(b). Petitioner submits that, for the purposes of this proceeding and the grounds presented herein, no claim term requires express construction. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017).

While an express construction is not necessary for the purposes of this proceeding, the ’607 specification provides an example of the claim term “*definition of a service*” and that is relevant to the prior art discussed below: “The services can be defined through the use of catalog data such as a **service name**,... **description**,... **availability**, and the like.” Ex.1001, 15:65-16:1. Therefore, the ’607 patent describes a service name, description, and availability as examples of “*a definition of a service*.” Ex.1001, 15:65-16:1; Ex.1003, ¶44.

Also, while an express construction is not necessary for the purposes of this proceeding, the ’607 patent specification also provides an example of the claim term “*definition of... a plan*” that is relevant to the prior art below: “The plans

comprise information regarding pricing, features, and/or **service limits associated with the service** offered by the application service provider 160.” Ex.1001, 16:5-8. Therefore, the ’607 patent describes service limits associated with the service is an example of “*a definition of [] a plan.*” Ex.1001, 16:5-8; Ex.1003, ¶44.

X. RELIEF REQUESTED AND THE REASONS FOR THE REQUESTED RELIEF

Petitioner asks that the Board institute an IPR trial and cancel the Challenged Claims.

XI. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

A. Challenged claims

All of the Challenged Claims are asserted in *Zilkr Cloud Technologies, LLC v. Cisco Systems, Inc.* No. 2-22-cv-00166 (E.D. Tex. filed May 20, 2022). Ex.1020. A finding that the Challenged Claims are unpatentable in this proceeding will resolve the parties’ dispute and obviate the need for a trial regarding the ’607 patent in the co-pending litigation, substantially reducing the time and expense of that litigation for all parties.

B. Statutory grounds for challenges

Ground	Claims	Basis
#1	1-3 and 5-6	35 U.S.C. § 103 over de Castro in view of Robb

U.S. Pat. Pub. No. 2012/0016932 to de Castro et al. (Ex.1005, “de Castro”)

published on January 19, 2012.

U.S. Pat. No. 8,126,722 to Robb (Ex.1006, “Robb”) issued on February 28, 2012.

De Castro and Robb are prior art under 35 U.S.C. § 102(a)(1).¹

Petitioner’s obviousness Ground relies on the combined teachings of the references as understood by a POSITA and not on a physical incorporation of elements from one reference into another. *See In re Mouttet*, 686 F.3d 1322, 1332 (Fed. Cir. 2012).

Petitioner cites to additional prior art as evidence of the background knowledge of a POSITA and to provide contemporaneous context to support Petitioner’s assertions regarding what a POSITA would have understood from the prior art in the grounds. Ex.1009 is an XMPP Specification document referred by de Castro as the Rayo Specification. Its prior art status is confirmed by Ex.1013 and Ex.1016. Ex.1007 and Ex.1008 are web pages from Zapier.com. The prior art status of Ex.1007 and Ex.1008 is confirmed by Ex.1014 and Ex.1015. Ex.1024 and Ex.1025 are technical dictionaries for telecom and computer terms. The prior art status of Ex.1024 is confirmed by Ex.1026. The prior art status of Ex.1025 is

¹ Petitioner notes that the “Swear Behind” provisions of 37 C.F.R. § 1.131 do not apply to the ’607 patent. *See* MPEP §§ 715, 2159.02.

confirmed by Ex.1027.

The Petition’s use of supporting tertiary evidence has been repeatedly approved. *See Yeda Research v. Mylan Pharm. Inc.*, 906 F.3d 1031, 1041-1042 (Fed. Cir. 2018) (affirming the use of “supporting evidence relied upon to support the challenge”); 37 C.F.R. § 42.104(b); *see also K/S HIMPP v. Hear-Wear Techs., LLC*, 751 F.3d 1362, 1365-66 (Fed. Cir. 2014); *Arendi S.A.R.L. v. Apple Inc.*, 832 F.3d 1355, 1363 (Fed. Cir. 2016).

C. Ground 1: Claims 1-3 and 5-6 are obvious under 35 U.S.C. § 103 over de Castro and Robb

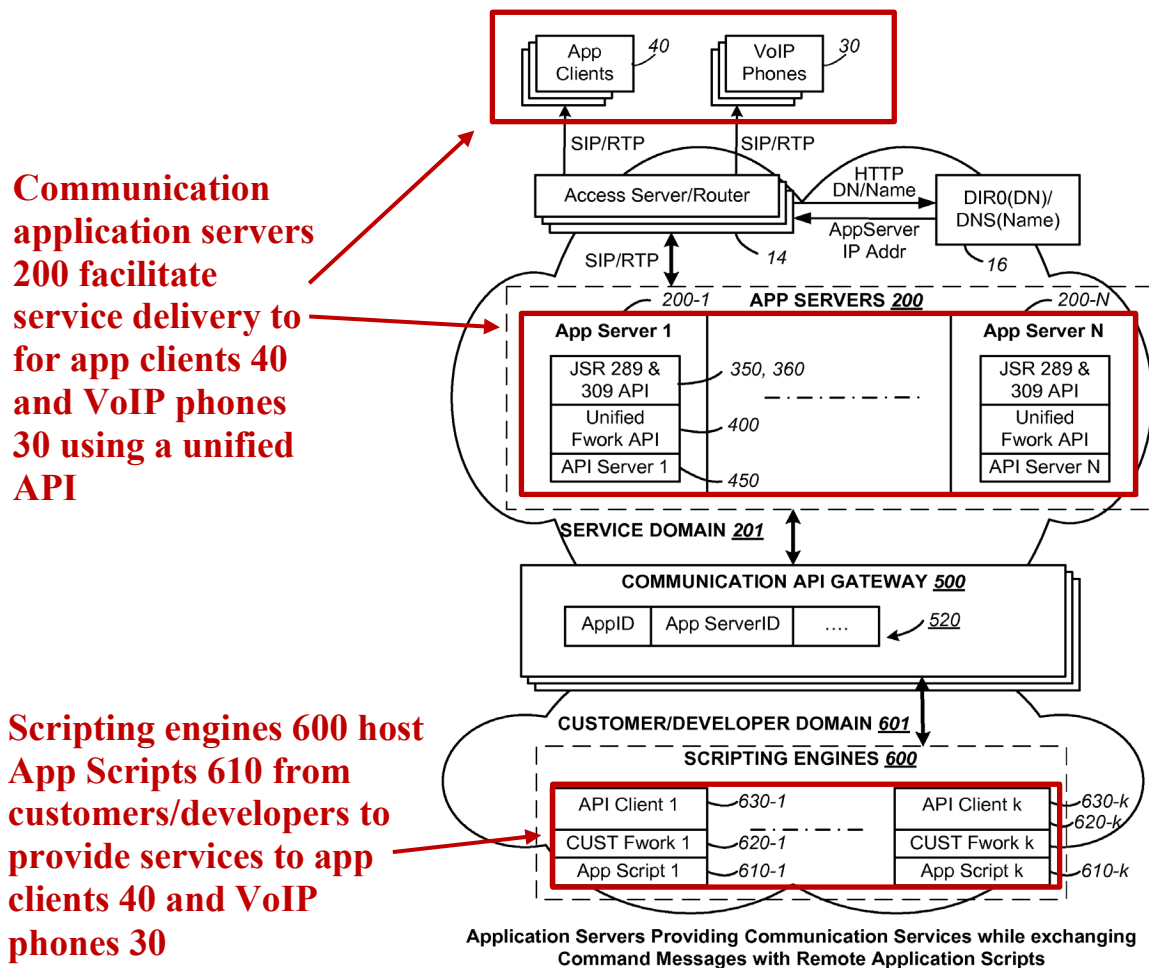
1. De Castro

De Castro describes “a plurality of communication application servers 200” that “allow[] application clients 40 ... to access the application hosted in the communication application servers.” Ex.1005, [0115]. A “client” includes a user device utilizing a telephony service, such as a phone. Ex.1005, [0145], [0116]; Ex.1003, ¶48.

De Castro’s communication application servers communicate with “a plurality of scripting engines 600” that are used to “host and process the customers’ applications scripts 610.” Ex.1005, [0119]. Note that De Castro carefully distinguishes the terms “customer” and “client.” Whereas “client” refers to an end user who uses a service (or to an end user’s device), the term “customer”

refers a developer that provides a service by coding an application script. Ex.1005, [0039], [0041], [0113]-[0114]; Ex.1003, ¶49.

Fig.14 below illustrates VoIP phones 30 and application clients 40 connected to scripting engines 600 via communication application servers 200. Ex.1003, ¶50.



Ex.1005, Fig.14 (annotated); Ex.1003, ¶¶50-51

The communication application servers 200 use “a messaging API protocol

[that] enables an application script to pass commands remotely to the communication application server to operate it.” Ex.1005, [0038]. This messaging API protocol or “unified API” allows a “customer using the communication services [to] code the application script in a preferred programming language using a custom framework providing a set of customer-specific libraries.” Ex.1005, [0038], [0041]. An example of the messaging API protocol is the “Rayo protocol” which “primarily deals with calls” and telecommunications services associated with a user’s telephone number. Ex.1005, [0138]-[0141]; Ex.1003, ¶¶50-52.

De Castro is analogous art to the ’607 patent and is from the same field of endeavor, providing communications services. Ex.1005, [0002]; Ex.1001, 1:23-26; Ex.1003, ¶53. For example, de Castro describes a “unified framework” that is used to provide application clients access to a variety of communication applications. Ex.1005, Abstract. Similarly, the ’607 patent provides a “unified services platform” to access “a variety of services and applications from multiple different ecosystems.” Ex.1001, 13:60-63, 9:60-62; Ex.1003, ¶53. Both de Castro and the ’607 patent describe using a common API protocol for the various service providers to execute their communication applications. *See* Ex.1001, Abstract; Ex.1005, Abstract; Ex.1003, ¶53. De Castro is also reasonably pertinent to the particular problem of the ’607 patent. Ex.1003, ¶53. For example, de Castro describes how service providers may use their own APIs to provide

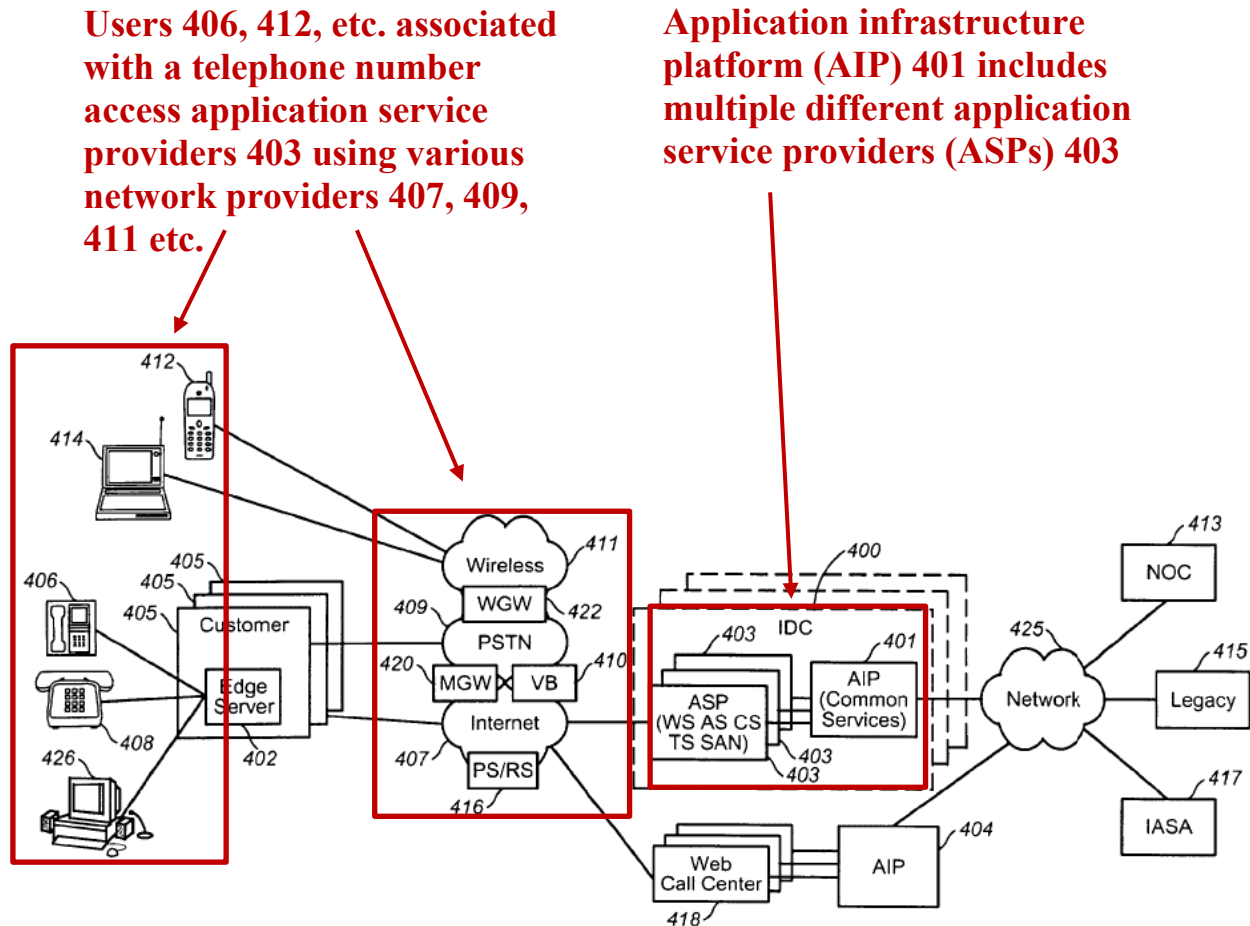
communications services to clients by telephone. *See* Ex.1001, Abstract, 2:1-5, 4:13-17, 11:33-42; Ex.1005, Abstract, [0139]-[0142]; Ex.1003, ¶53.

2. Robb

Robb relates to delivering “services and data to end user devices” such as telephones and personal digital assistants. Ex.1006, Fig. 5, Abstract, 9:43-65, 12:16-24, 14:40-48, 16:25-36. Robb describes “an application infrastructure platform (AIP) ... including multiple [application service provider] ASP systems providing application services to customers” (e.g., users²) who are associated with telephone numbers. Ex.1006, Abstract, Fig.5; Ex.1003, ¶54.

Fig. 5 depicts telephone users (e.g., 406 and 412) accessing AIP 401 that includes multiple different ASPs 403 via different network providers (e.g., Internet 407, and public switched telephone network 409). Ex.1006, Fig. 5, 15:56-16:9, 16:31-33; Ex.1003, ¶55.

² Note that in contrast to de Castro, Robb uses the term “customer” to refer to an end user who accesses telephony services. Ex.1006, 2:48-49.



Ex.1006, Fig.5 (annotated); Ex.1003, ¶¶55-56

Robb teaches that “the AIP 114 provides a collection of applications and services to the hosted ASPs 102.” Ex.1006, 7:29-30. To make this collection of applications available to other ASPs, Robb explains that an AIP may “aggregate its own applications and services with those of both external ASPs/Internet Service Vendors (ISVs) and its own ASPs into a ‘Product Catalog’.” Ex.1006, 7:53-55. ASPs can then select from the cataloged applications and services to “repackage and resell to their customers as corporate portals.” Ex.1006, 7:30-32. Robb

explains that this service aggregation and customization technique “enables service providers to leverage more customers by wholesaling or leasing to other service providers than they can address on their own.” Ex.1006, 26:47-49. Robb refers to the selected and customized services in the customer specific portals as “service packages,” where a service package includes “an aggregation of services for a user role” and “SLA [service-level agreement] terms and conditions to the service.” Ex.1006, 31:40-41, 34:16-17, 34:25-27; Ex.1003, ¶¶56-58.

Robb is analogous art to the '607 patent and is from the same field of endeavor, providing communications services. Ex.1006, 3:40-42; Ex.1001, 1:23-26; Ex.1003, ¶59. For example, Robb describes a “application infrastructure platform” that is used to provide application clients access to “multiple ASP systems providing application services to customers.” Ex.1006, Abstract. Similarly, the '607 patent provides a “unified services platform” to access “a variety of services and applications from multiple different ecosystems.” Ex.1001, 13:60-63, 9:60-62; Ex.1003, ¶59. Both Robb and the '607 patent describe using a common API protocol or platform for the various service providers to execute their communication applications. *See* Ex.1001, Abstract; Ex.1006, 15:56-16:9; Ex.1003, ¶59. Robb is also reasonably pertinent to the particular problem of the '607 patent. For example, Robb describes how application service providers may integrate their services into a platform to provide communications services to

clients by telephone. *See* Ex.1001, Abstract, 11:33-42; Ex.1006, 6:25-36; Ex.1003, ¶59. Robb also describes how service providers may select from cataloged services and publish a selection of services on a portal for telephone clients. *See* Ex.1001, 12:9-16; Ex.1006, 7:30-32, 7:53-55, 34:16-17; Ex.1003, ¶59.

3. Reasons to combine de Castro and Robb

A POSITA considering the teachings of de Castro would have also considered the teachings of Robb, as they are analogous art and pertain to the same field of endeavor. *See, e.g.*, Ex.1005, [0039]; Ex.1006, Fig.5; Ex.1003, ¶60. Each of these references describes integrating multiple telecommunications application services and network providers for end users associated with a telephone number. *See, e.g.*, Ex.1005, Fig.14, [0139]-[0141]; Ex.1006, Fig.2, Fig.5, 9:43-50; Ex.1003, ¶60.

It would have been obvious, beneficial, and predictable to utilize Robb's teachings regarding aggregating telecommunication services in a product catalog and repackaging and reselling the cataloged services for use on corporate portals in combination with de Castro's system. Ex.1003, ¶61. For example, a POSITA would have appreciated the benefits of deploying various tailored combinations of de Castro's Rayo-based services using Robb's product catalog and corporate portal techniques. Ex.1003, ¶61. Indeed, de Castro already contemplates: (1) developers coding application scripts; (2) these scripts including commands that are expressed

in standardized Rayo protocol messages; and (3) simplifying the telecommunications application development process to allow the developer to “focus on the business logic.” Ex.1005, [0038], [0041], [0098]; Ex.1003, ¶62. For instance, in de Castro’s system, developers can “code the application script in a preferred programming language,” which is then “expressed in terms of the finite set of commands” of a unified API, e.g., the Rayo protocol. Ex.1005, [0041], [0138]-[0141]; Ex.1003, ¶62. De Castro also teaches that the unified API (Rayo protocol) “allows application clients 40 such as, for example, VoIP phones to access the applications hosted in the communication application servers.” Ex.1005, [0115]; Ex.1009, Ex.1003, ¶¶62-64.

Robb provides details on how a platform that aggregates multiple telecommunications services from different providers (like de Castro’s communication application server) publishes the aggregated services to a product catalog for selection and customization by other service providers. Ex.1003, ¶65. Robb also explains how service providers use a “corporate portal” to leverage the aggregation of services in the product catalog to customize, repackage, and resell service combinations to their end users Ex.1006, 26:47-49, 34:16-17, 34:25-27, 35:9-10; Ex.1003, ¶65. For instance, Robb describes publishing “service packs” (i.e., application service software code and licensing terms) on a corporate portal. Ex.1006, 34:16-17, 34:25-27; Ex.1003, ¶65. As described in more detail in sections

XI.C.4.[1.9] and [2.1], the “product catalog” is the equivalent of the claimed “*provisioning catalog*” of [1.9]. The “corporate portal” is the equivalent of the claimed “*services catalog*” of [2.1]. A POSITA would have found the combination obvious, where Rayo-protocol services are first made available to other developers via a product catalog, and then customized and repackaged for de Castro’s clients via a portal website educating users about the available services, per Robb. Ex.1003, ¶65.

A POSITA would have recognized the benefits of publishing the different application services provided by (or available using) de Castro’s system in a product catalog, per Robb. *See* Ex.1005, Fig.14, [0041], [0138]-[0141]; Ex.1006, Fig.5, 34:16-17, 34:25-27; Ex.1003, ¶66. For instance, a POSITA would have recognized that the combination beneficially allows different developers or service providers to “leverage more customers by wholesaling or leasing to other service providers than they can address on their own.” Ex.1006, 26:48-49; Ex.1003, ¶66. Further, a POSITA would have recognized the advantages of using Robb’s product catalog to enable service providers to “provide customized content to the end user” by repackaging selected services from the product catalog for use through a customer-specific portal. Ex.1006, 9:66-67; Ex.1003, ¶66. This advantageously expands the range of telecommunication services and service packages available to both service providers and end users. Ex.1006, 6:28-32; Ex.1003, ¶66. The

combination thus beneficially allows service providers “to augment their own offerings presented as blended services through a unified corporate portal.”

Ex.1006, 6:32-35; Ex.1003, ¶66.

A POSITA also would have recognized that using a corporate portal that publishes service packs, per Robb, would be beneficial for de Castro’s system because the “services offered through a corporate portal have also grown more robust.” Ex.1006, 6:53-54; Ex.1003, ¶67. For instance, Robb explains that “[i]nstead of purchasing licenses (seats) for employees to use various software packages that are installed on their personal computers, many corporations find it less burdensome to lease software on a per use basis.” Ex.1006, 6:53-57; Ex.1003, ¶67. Robb’s technique thus beneficially provides de Castro’s system a flexible and user-friendly implementation—because as Robb explains, “portals can have a specific focus, such as personal portals, business-to-business (B2B) portals, business-to-employee (B2E) portals, and wireless portals. Portals filter, aggregate, and present the data to the end user.” Ex.1006, 10:1-5; Ex.1003, ¶67.

The combination of de Castro and Robb is merely the ordinary use of common techniques (aggregating services in a product catalog and publishing services on a portal, per Robb) in the similar unified communication application system of de Castro to yield predictable results (Rayo-protocol application services available in a product catalog and a portal). Ex.1003, ¶68. Robb’s product catalog

and portal techniques were also specifically intended to be implemented in servers integrating multiple telecommunications applications (such as de Castro's). *See* Ex.1005, Fig.14, [0038], [0041]; Ex.1006, Fig.5, Fig.16, 27:50-58, 28:4-17, 34:16-17, 34:25-27; Ex.1003, ¶68. All of these reasons, individually and collectively, would have motivated a POSITA to combine the references to arrive at the invention recited in the Challenged Claims. Ex.1003, ¶68.

The results would have been predictable and there would have been a reasonable expectation of success in the combination given that both de Castro and Robb describe similar concepts (e.g., telephone users accessing different application services over telecommunications and internet networks). *See* Ex.1005, Fig.14; Ex.1006, Fig.5; Ex.1003, ¶69. Furthermore, aggregating telecommunications services in a product catalog and publishing services in a corporate portal was long known in the art, thus confirming predictability and expectation of success. Ex.1003, ¶70 (citing Ex.1007, Ex.1008). Any modification of de Castro's teachings, including implementing software programming or hardware, to accommodate the teachings of Robb, would have been within the level of ordinary skill in the art. Ex.1003, ¶71. For instance, a POSITA would have understood computer programming techniques and web design for publishing application services (e.g. Rayo protocol services) on a corporate portal. Ex.1003, ¶71.

4. Claim 1

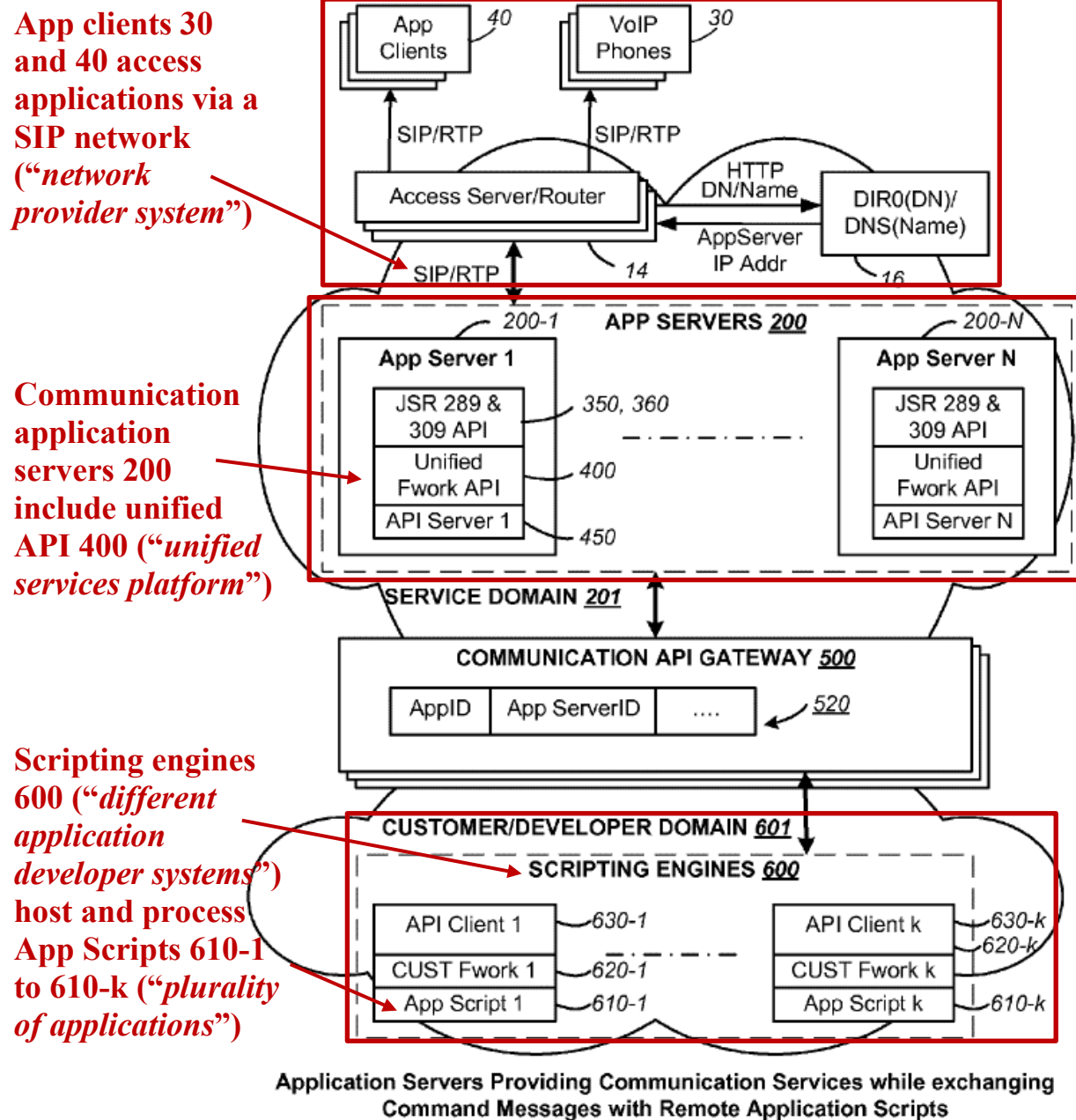
[1.0] *A method of provisioning a plurality of applications from different application developer systems to a network provider system through a unified services platform, the method comprising:*

De Castro describes providing “telephony and communication services” via a “**unified framework**” (“*unified services platform*”) that “supports a **unified API**.” Ex.1005, [0038]-[0039]. The unified framework facilitates the deployment and execution of “**application scripts**” (“*provisioning a plurality of applications*”) for clients over a network. Ex.1005, Abstract, [0039], [0119]; Ex.1003, ¶72.

These application scripts are provided by a plurality of “developers” or “customers with application scripts.” Ex.1005, [0035], [0039]. The “application scripts [are] processed remotely by **appropriate scripting engines**.” Ex.1005, [0038]. These scripting engines are “**third party scripting engines**” associated with the plurality of third-party developers (“*different application developer systems*”). Ex.1005, [0119], [0041]; Ex.1003, ¶73.

The unified framework “allows application clients ... to access the application hosted in the communication application servers.” Ex.1005, [0115]. Clients access the telecommunications application services (e.g., services for controlling phone calls) hosted in the communication application servers “**via SIP**” networks (“*network provider system*”). Ex.1005, [0139]; Ex.1003, ¶74. This arrangement is depicted in Fig.14 where App Scripts 610-1 to 610-k

(“*applications*”) are provided to SIP-network clients 40 and 30 via communication application servers 200. Ex.1005, Fig.14. The figure also illustrates that communication application servers 200 include unified API 400 (“*unified services platform*”). Ex.1005, Fig.14; Ex.1003, ¶¶74-75.



Ex.1005, Fig.14 (annotated); Ex.1003, ¶¶75-76

Thus, to the extent the preamble is limiting, de Castro teaches developer applications, which are processed by scripting engines, providing services to clients on a SIP network via communication application servers via a unified API.

This discloses the claimed *method*. Ex.1003, ¶76. As demonstrated below, the combination of de Castro and Robb discloses a method “*comprising*” the recited elements.

[1.1] *establishing a plurality of service accounts for a corresponding plurality of application service provider systems,*

First, de Castro teaches a plurality of “**scripting engines**” (“*plurality of application service provider systems*”) for “application scripts to be processed remotely.” Ex.1005, [0038], Fig.14; Ex.1003, ¶77.

Second, each scripting engine that provides application services is identified by an “**AppID**.” Ex.1005, Fig.14, Fig.19. Information about these services is registered in a “**registry**.” Ex.1005, Fig.14, Fig.19, [0130]; Ex.1003, ¶78.

De Castro describes this registration process in more detail, including that “**registry 520...registers...the customers’ applications in the customer/developer domain.**” Ex.1005, [0130]. Within registry 520, also “logged are ... **the properties and status associated with each API clients 630.**” Ex.1005, [0130]. In particular, each scripting engine includes an API client 630, which “interprets [each App Script] and renders it in terms of the commands of the finite set” of “commands supported by the unified framework API 400.” Ex.1005, [0119], [0127]; Ex.1003, ¶79.

Fig.19 below illustrates registry 520 that logs information including the

“properties/status” about each scripting engine’s API client 630 identified by an “App ID.” Ex.1005, Fig.19, [0130]; Ex.1003, ¶80.

Registry 520 registers and logs information from each application service scripting engine (App ID) (“*plurality of service accounts*”) including application properties and status information

Registry 520 

SERVICE DOMAIN			USER DOMAIN		
App ServerID	App Server Properties/ Status		App ID	App Properties/ Status	

Ex.1005, Fig.19 (annotated); Ex.1003, ¶80

A POSITA would have understood that the information stored in the registry for each service is a “*service account[]*.” Ex.1003, ¶81. Registering each of the application services offered by the scripting engines renders obvious “*establishing a plurality of service accounts.*” Ex.1003, ¶¶81-82.

Thus, de Castro teaches a registry that registers and logs service-specific information (“*service accounts*”) about the scripting engine of each application service (“*application service provider systems*”), including logging application properties and status information, which renders obvious “*establishing a plurality of service accounts for a corresponding plurality of application service provider systems.*” Ex.1003, ¶82.

[1.2] wherein the plurality of application service provider systems provide services for use in association with a telephone number of a subscriber of telecommunication services;

First, as discussed in [1.1], de Castro’s system includes “**scripting engines**” (“*plurality of application service provider systems*”) that process application scripts to “**provide telephony and communication services**” (“*provide services*”). Ex.1005, [0039], Fig.14; Ex.1003, ¶82.

Second, the scripting engines are used to control calls made to and from the telephone numbers of a clients. Ex.1005, [0141]-[0142]; Ex.1003, ¶¶83-84.

De Castro explains that a scripting engine provides a “**telephony-enabled service**,” such as a service for “application clients such as **VoIP phones 30**” or “**application clients 40 and other terminal equipment.**” Ex.1005, [0002], [0116], [0102]; Ex.1003, ¶84. A POSITA would have found it obvious that each of the described application clients is an example of the claimed “*subscriber of telecommunication services.*” Ex.1003, ¶84. As examples of telephony services hosted by the scripting engines, de Castro describes services for controlling “Incoming Calls” and Outbound Calls” made to and from the phone numbers of clients. Ex.1005, [0138]-[0142]. Thus, the telephony services are provided in “*association with a telephone number of a subscriber.*” Ex.1003, ¶84.

De Castro provides example messages showing that the provided services are associated with a telephone number of a subscriber. Ex.1005, [0138]-[0142].

For example, an incoming call results in a Rayo protocol “offer” message that identifies the telephone numbers of the calling subscriber (“from”) and the called subscriber (“to”):

<!-- Message comes from the Call’s JID -->

...

<offer xmlns=‘urn:xmpp:Rayo:1’

to=‘tel:+18003211212’

from=‘tel:+13058881212’>

Ex.1005, [0141]. Similarly, de Castro describes an example Rayo protocol message for making an Outbound Call. The message identifies the telephone numbers of the calling subscriber (“from”) and the called subscriber (“to”):

<dial **to=‘tel:+13055195825’ from=‘tel:+14152226789’**

xmlns=‘urn:xmpp:Rayo:1’>

Ex.1005, [0142]; Ex.1003, ¶¶85-86.

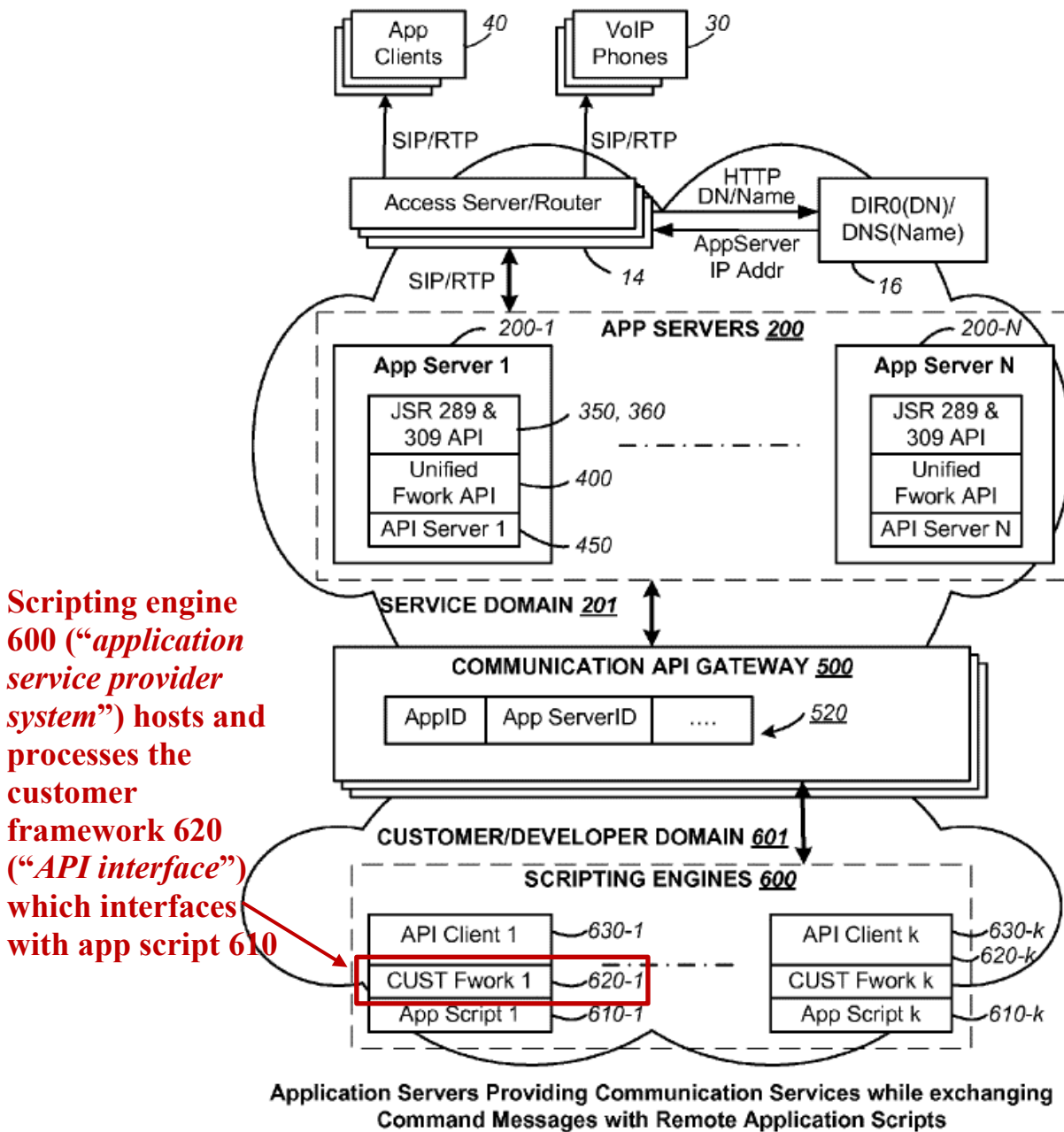
Thus, de Castro teaches scripting engines processing Rayo messages to control calls to and from the phone numbers of telephone-service clients, which renders obvious “*wherein the plurality of application service provider systems provide services for use in association with a telephone number of a subscriber of telecommunication services.*” Ex.1003, ¶87.

[1.3] receiving, by a processor system, a plurality of API interfaces from the plurality of application service provider systems, wherein at least two of the API interfaces are different,

First, de Castro teaches that each scripting engine (“*application service provider system*”) includes a “**custom framework**” to interface with an application script. Ex.1005, [0041], [0119], [0067]; Ex.1003, ¶88.

De Castro explains that a developer “can code the application script in a preferred programming language using a **custom framework providing a set of customer-specific libraries.**” Ex.1005, [0041]. Each of these custom “frameworks” is an “API” (“*a plurality of API interfaces*”): “**Frameworks** are a special case of software libraries in that they are reusable abstractions of code wrapped in a **well-defined API.**” Ex.1005, [0067]. The custom framework API interfaces with the application script, as the “**processing of App Script 1 is facilitated by a customer-specific framework 620**, such as CUST Fwork 1 620-1 which supplies a set of customer-specific libraries.” Ex.1005, [0119]; Ex.1003, ¶89.

Fig.14 below illustrates scripting engine 600-1 hosting and processing the customer-specific framework 620-1 API, which interfaces with app script 610-1. Ex.1005, Fig.14, [0119].



Ex.1005, Fig.14 (annotated); Ex.1003, ¶¶90-91

Second, de Castro describes the application scripts and customer framework APIs as being “hosted” by the scripting engines. Ex.1005, [0039], [0110], [0115], [0119]. For instance, Fig.14 shown above illustrates customer framework APIs

(CUST Fwork 620-1 to 620-k) hosted in scripting engines (600-1 to 600-k).

Because the customer framework APIs are illustrated as being located within and hosted by the scripting engines, a POSITA would have understood that the customer framework APIs were “*receiv[ed]*” by the scripting engines. Ex.1003, ¶¶91-92. That is, scripting engines have customer frameworks implementing customer-specific APIs; and the scripting engines, at some point, must have “*receiv[ed]*” the customer frameworks. Ex.1003, ¶92.

De Castro also explains that application scripts and customer framework APIs are “**processed remotely** by appropriate scripting engines.” Ex.1005, Fig.14, [0038]. A POSITA would have understood this disclosure to mean that the scripting engines use at least one computer processor (“*a processor system*”). Ex.1003, ¶93. For instance, De Castro describes how a customer framework API facilitates processing by a scripting engine: “**processing of App Script 1 is facilitated by a customer-specific framework 620, such as CUST Fwork 1 620-1** which supplies a set of customer-specific libraries.” Ex.1005, [0119]. Within the scripting engine, “**processing is performed** by an API client 630 such as API client 1 630-1 which interprets App Script 1 and renders it in terms of the commands of the finite set defined at the API server 450 of the communication application server 200.” Ex.1005, [0119]. Therefore, the scripting engines’ hosting and processing of customer framework APIs renders obvious “*receiving, by a*

processor system, a plurality of API interfaces.” See Ex.1005, Fig.14, [0038], [0119]; Ex.1003, ¶93.

Third, de Castro teaches that its various customer framework APIs are different (“*at least two of the API interfaces are different*”). Ex.1005, [0039]. As discussed above, the application script can be coded in a **“preferred programming language using a custom framework providing a set of customer-specific libraries.”** Ex.1005, [0041]. For instance, these languages “include Ruby, Python, Groovy and PHP.” Ex.1005, [0035]. Because the custom frameworks (“*API interfaces*”) are customer specific and interface with applications written in different languages, it would have been obvious to a POSITA for the custom framework APIs to be different from one another. Ex.1005, [0067], [0041]; Ex.1011, 2:65-3:6; Ex.1003, ¶¶94-95.

Thus, de Castro teaches scripting engines hosting and processing customer framework APIs that interface with application scripts coded in different languages, which renders obvious “*receiving, by a processor system, a plurality of API interfaces from the plurality of application service provider systems, wherein at least two of the API interfaces are different.*” Ex.1003, ¶96.

[1.4] *integrating, by the processor system, each of the plurality of API interfaces with a unified services platform API interface;*

First, each “communication application server is provided with a **unified**

framework” where “[t]he framework supports a unified API having class objects and functions conforming to a telephony object model.” Ex.1005, [0109], [0069]. In particular, the “**unified framework provides a unified API that gives access to a set of unified class objects** for call control and media control.”

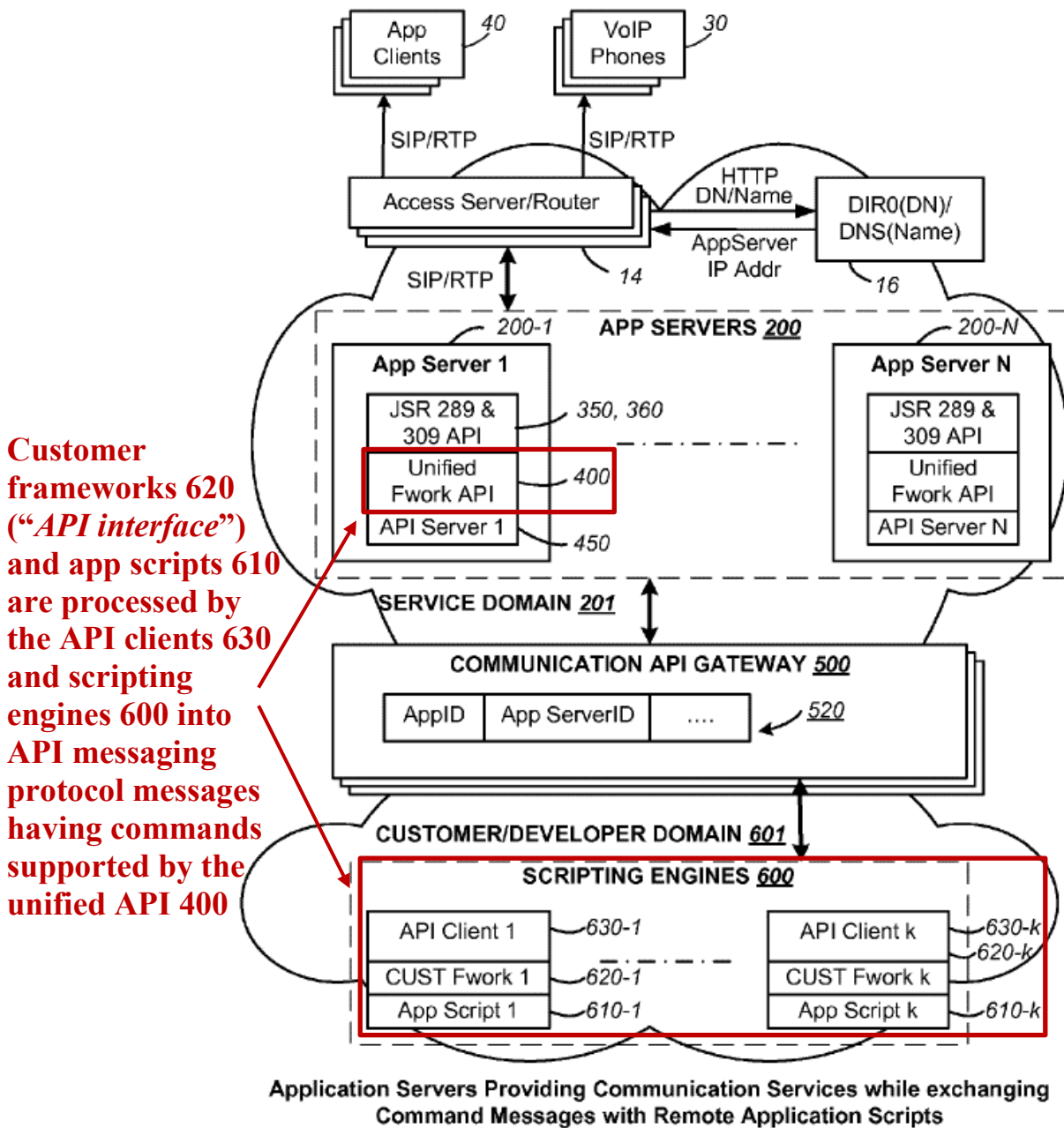
Ex.1005, [0100]. Therefore, “the communication application server 200 does not proceed to retrieve a script for the specific communication application,” but instead “it **exposes the unified communication framework API 400 as a finite set of commands.**” Ex.1005, [0118]. De Castro’s unified API is the claimed “*unified services platform API interface.*” Ex.1003, ¶¶97-98.

Second, de Castro’s scripting engines include at least one computer processor (“*processor system*”) as discussed at [1.3]. Each of the scripting engines “**interprets**” or processes the customer frameworks (“*API interfaces*”) and application scripts. Ex.1005, [0119]; Ex.1003, ¶99.

For instance, scripting engines 600 include “API client 630 [which] **interprets App Script 1 and renders it in terms of the commands of the finite set defined at the API server.**” Ex.1005, [0119]. Next, the “API client 630 then **packages the commands as messages** in accordance with the messaging API protocol.” Ex.1005, [0119]. In this way, the application scripts and customer frameworks hosted by the scripting engines use “a messaging API protocol” to communicate with “the communication application server on the network to

execute their applications as **a series of these commands**” expressed in the unified API. Ex.1005, [0118]; Ex.1003, ¶¶100-101.

Fig.14 illustrates scripting engines 600 and API clients 630 that process the application scripts 610 and customer frameworks 620 into a finite set of commands supported by unified API 400. Ex.1005, Fig.14; Ex.1003, ¶102.



Ex.1005, Fig.14 (annotated); Ex.1003, ¶102

The scripting engines process application scripts into “conformity to the API messaging protocol” (“integrating”), which uses “embedded command[s] from the finite set of commands supported by the unified framework API 400”

(“*integrating with a unified services platform API interface*”). Ex.1005, [0127]; Ex.1003, ¶103.

A POSITA would have found it obvious that de Castro’s system would “*integrat[e]*” the “*API interfaces*,” because integrating telecommunications services and their associated APIs was well-known in the art. *See* Ex.1006, 3:31-39; Ex.1010, Fig.2, 5:31-58; Ex.1012, 37:5-9, 458:5-22; Ex.1003, ¶104.

Thus, de Castro teaches scripting engines that process customer framework APIs into the commands of a unified API, which renders obvious “*integrating, by the processor system, each of the plurality of API interfaces with a unified services platform API interface.*” Ex.1003, ¶105.

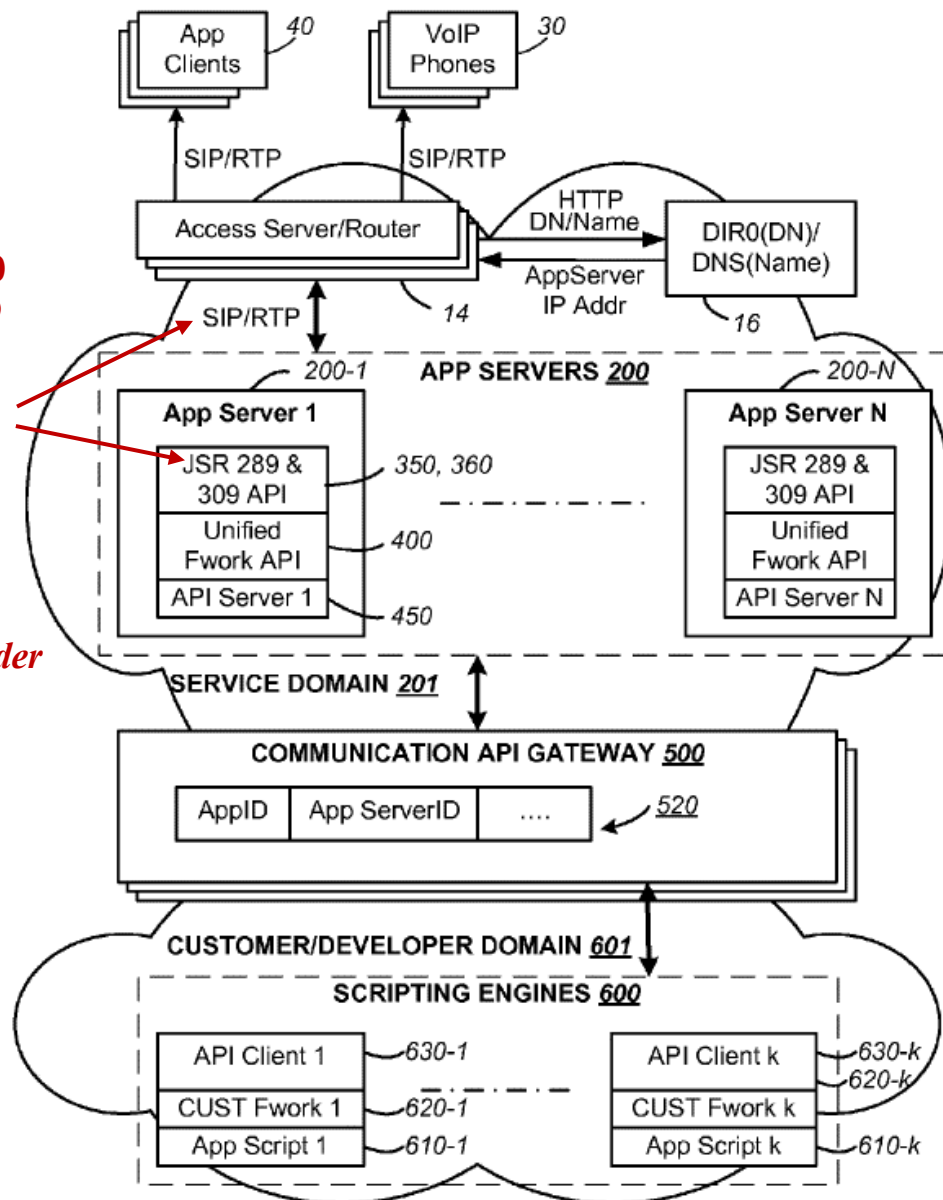
[1.5] *receiving a provider API interface for a network provider system;*

First, as discussed at [1.0], clients access the call-related application services “**via SIP**” networks (“*network provider system*”) connected to the application communication servers. Ex.1005, [0139], Fig.14; Ex.1003, ¶106.

Second, de Castro explains that the application communication servers 200 include a “**SIP Servlet (JSR 289) API 350**” (“*a provider API interface*”). Ex.1005, [0070]. Fig.14 below illustrates that SIP API (JSR 289) 350 executes on communication application server 200 for interfacing with a SIP network. Ex.1005, Fig.14, [0070], [0139]. Because the SIP API 350 executes on server 200, it would have been obvious to a POSITA for the SIP API to be loaded onto or “*receiv[ed]*”

by server 200 prior to execution. *See* Ex.1005, Fig.14, [0070], [0139]; Ex.1003, ¶107. Figure 14 illustrates SIP API 350 as being included within, and thus already “receiv[ed]” by, server 200. Ex.1005, Fig.14.

App servers 200 receive JSR 289 API (e.g., SIP API) (“provider API interface”) for interfacing with a client-facing SIP network (“network provider svstem”)



Application Servers Providing Communication Services while exchanging Command Messages with Remote Application Scripts

Ex.1005, Fig.14 (annotated); Ex.1003, ¶107

Thus, de Castro teaches communication application servers including a SIP API for interfacing with a SIP network, which renders obvious “*receiving a provider API interface for a network provider system.*” Ex.1003, ¶108.

[1.6] *integrating the provider API interface with the unified services platform API interface, wherein at least one of the plurality of API interfaces from the plurality of application service provider systems is not compatible with the provider API interface;*

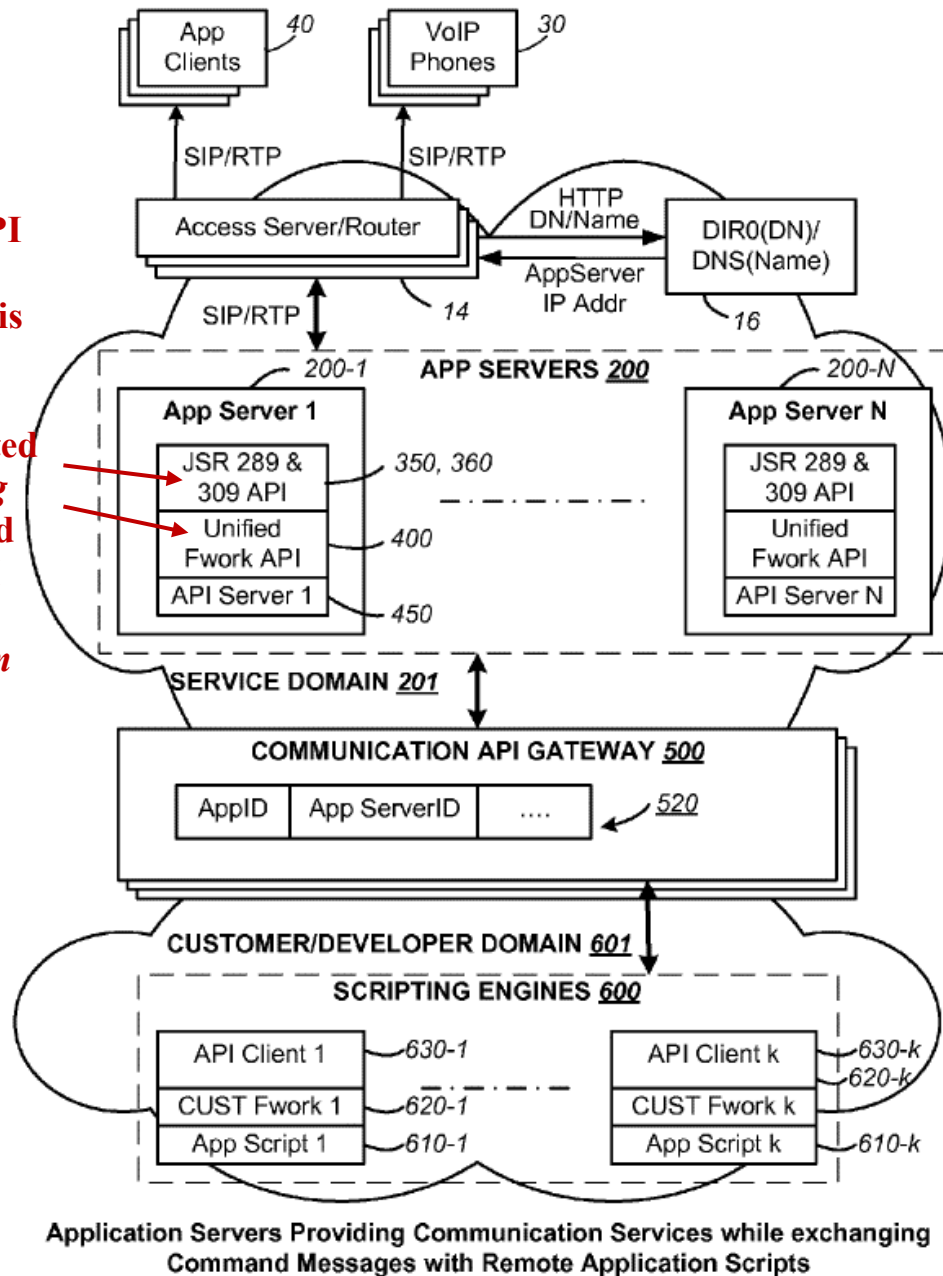
First, de Castro teaches that, within the communication application servers, the “JSR 289[] API 350” (e.g., SIP API) is a programming interface that “deal[s] with high-level objects abstracted from low-level” SIP network communication objects, for use by the unified framework AIP 400. Ex.1005, [0070], [0074], Fig.14; Ex.1003, ¶109.

The “unified communication framework provides high-level, unified objects built from lower-level object of JSR 289/309”. Ex.1005, [0096]. Therefore, “[b]y dealing with **high-level objects abstracted from low-level objects of JSR 289[]**, programming at the application level is more efficient and relevant to the problems at hand.” Ex.1005, [0074]; Ex.1003, ¶110.

Figure 14 depicts JSR 289 (SIP API 350) interfacing with unified framework API 400. Ex.1005, Fig.14. At this interface, JSR 289 abstracts and builds low-level call objects into high-level objects for use by unified framework API 400, which a POSITA would have understood to be the claimed “*integrating the provider API*

interface with the unified services platform API interface.” Ex.1005, Fig.14,
[0096]; Ex.1003, ¶111.

JSR 289 SIP API 350 (“provider API interface”) is interfaced with and provides objects abstracted for (“integrating ... with”) unified framework API 400 (“unified services platform API interface”)



Ex.1005, Fig.14 (annotated); Ex.1003, ¶111

Second, de Castro’s SIP networks are associated with a “SIP API” or JSR

289 having low-level call objects. De Castro's application services that utilize the SIP networks use a "**customer-specific framework**" API that abstracts various programming languages such as Ruby or Python. Ex.1005, [0067], [0106], [0119]; Ex.1003, ¶112.

For instance, "**SIP servlet (JSR 289) for call control ... provides a Java API**" ("*provider API interface*"), where Java is a type of programming language. Ex.1005, [0026]-[0027]. "SIP servlets interact with clients by **responding to incoming SIP requests and returning corresponding SIP responses.**" Ex.1005, [0026]. The JSR 289 SIP API, therefore, deals with "**low level call control objects**" of an SIP network. Ex.1005, [0030], [0070]; Ex.1003, ¶113.

In contrast to JSR 289, which is an API for SIP network call objects, each of the customer-specific framework APIs 620 ("*API interfaces*") are used for processing application scripts 610, which are written in different programming languages. Ex.1005, [0026], [0106], [0119]. De Castro explains that "application developers are using **other light-weight protocols and languages** to code the application scripts. These include **Ruby, Python, Groovy and PHP.**" Ex.1005, [0106]. Then, the application scripts, written in these various "**customer ... preferred programming language[s]**" are processed "**using a custom framework providing a set of customer-specific libraries.**" Ex.1005, [0106], [0041]; Ex.1003, ¶114.

Therefore, a POSITA would have understood that the Java-based JSR 289 SIP API 350 is not compatible with the API of an application script coded in, e.g., Ruby, TAPI, or Python. Ex.1005, [0070], [0106]; Ex.1024, p. 461; Ex.1025, pp. 723-727, 195; Ex.1003, ¶¶115-117. For instance, the SIP API is not natively compatible with the Rayo API, as Rayo messages cannot be parsed by a SIP server. Ex.1003, ¶115. These teachings from de Castro render obvious “*wherein at least one of the plurality of API interfaces from the plurality of application service provider systems is not compatible with the provider API interface.*” Ex.1003, ¶¶115-117.

Thus, de Castro teaches low-level call objects using a Java-based SIP API, which interfaces with and abstracts objects for a unified API, whereas application scripts coded in customer-preferred languages use customer-specific framework APIs, which renders obvious “*integrating the provider API interface with the unified services platform API interface, wherein at least one of the plurality of API interfaces from the plurality of application service provider systems is not compatible with the provider API interface.*” Ex.1003, ¶118.

[1.7] receiving a definition of a service and a plan for each of the plurality of application service provider systems;

First, de Castro’s “registry 520 [] registers ... the customers’ applications in the customer/developer domain 601.” Ex.1005, [0130]. Further, registry 520

also “**log[s] ... the properties and status associated with each API clients 630.**”

Ex.1005, [0130]. De Castro thus teaches that the registry logs information for each application and the API client of a scripting engine (“*service provider systems*”).

Ex.1005, [0130]. A POSITA would have understood that logging, at the registry 520, such information about the application scripts 610 and scripting engines 600, includes “*receiving*” and storing such information at the registry 520. Ex.1005, [0130]; Ex.1003, ¶119.

Because the registry includes “properties” information for each scripting engine (Ex.1005, [0130]), a POSITA would have found it obvious for one of the “properties” to be a short description or name of the application (each an example “*definition of a service*”). Ex.1003, ¶120. Storing a service’s name, description, or both, would facilitate an administrator in determining to which service the other, related properties pertain. Ex.1003, ¶120. The registry also logs “status” information for each of the applications (Ex.1005, [0130]), which a POSITA would have found obvious to include service availability information (another example “*definition of a service*”). Ex.1003, ¶120.

Therefore, a POSITA would have understood that these teachings from de Castro render obvious “*receiving a definition of a service ... for each of the plurality of service provider systems.*” Ex.1005, [0130]; Ex.1003, ¶121. As discussed above at Section IX, the ’607 patent describes a service name,

description, and availability as examples of “*a definition of a service.*” Ex.1001, 15:65-16:8; Ex.1003, ¶121. De Castro’s disclosure of the registry including application properties (e.g., service name) and status information is just like the examples of service name, description, and availability in the ’607 patent, and further confirm that de Castro renders obvious this limitation. Ex.1005, [0130]; Ex.1003, ¶121.

Second, Robb, like de Castro, describes “providing application services to customers” who use “SIP[] phones.” Ex.1006, Abstract, 9:49-50. Robb further describes an application infrastructure platform (AIP) that “provides a collection of applications and services to the hosted ASPs 102; the ASPs 102 in turn may repackage and resell to their customers as corporate portals.” Ex.1006, 7:29-32. Specifically, Robb describes aggregating applications and services from ASPs and ISVs “into a ‘**Product Catalog**,’” which “may be used by new ASPs to select their services for customization and resale to customers.” Ex.1006, 7:53-57, 17:63-67. “The aggregate product catalog enables service providers to leverage more customers by wholesaling or leasing to other service providers than they can address on their own.” Ex.1006, 26:47-49; Ex.1003, ¶¶122-123.

Robb teaches that the AIP also includes “**SLA [service-level agreement] Management**” which “provides for comparison of application or network events that affect delivery of applications and services to customers based on **SLA**

criteria.” Ex.1006, 26:67 – 27:5. A service pack created or defined by the ASPs includes “an aggregation of services for a user role” and “SLA [service-level agreement] terms and conditions to the service.” Ex.1006, 34:25-27. Robb teaches that, once an ASP has selected a service to add to a corporate portal, an AIP administrator “specifies the details of the Service Level Agreements (SLA) agreed by the AIP and the ASP.” Ex.1006, 33:39-42. As discussed above at Section IX, the ’607 patent describes service limits associated with the service—such as SLA terms and conditions—is an example “[*definition of*] a plan.” Ex.1001, 16:5-8; Ex.1003, ¶124.

A POSITA would have found it obvious to combine Robb’s teachings with de Castro’s system for the reasons provided above at Section XI.C.3. Ex.1003, ¶125. For instance, a POSITA would have been motivated to apply Robb’s product catalog technique to de Castro’s system, so that service providers (e.g., developers) would have access to a variety of applications and services. Ex.1003, ¶125. In the resulting combination, the properties and status of each application, such as a name or short description (“*definition of the service*”), as well as default service-level agreement (SLA) terms, conditions, and other SLA criteria (“[*definition of*] a plan”), would be included in information about the aggregated services that is published to or aggregated on the AIP product catalog for resale by ASPs.

Ex.1003, ¶¶125, 128. It would have been obvious to a POSITA that the services

added to the AIP's product catalog would include default SLA terms, the "details" of which would then be "specifie[d]" by the AIP administrator. Ex.1006, 33:39-42, 32:40-42; Ex.1003, ¶126. For instance, the prior art acknowledges that a common practice was to provide default or standard terms for a software service agreement that could be customized or "superseded" by a service provider. Ex.1022, table at cols. 3 and 4 (billing and customer service (BCS) providers specify "default terms" of a subscription); Ex.1023, [0058] (providers publish "standard SLA templates" and "describe the details of the various SLA templates). This approach would simplify and streamline the service package publishing process by allowing ASPs to use the default SLA terms and modify as necessary, rather than define from the SLA terms and criteria from scratch. Ex.1023, [0058] ("To expedite and/or simplify SLA negotiation, consumers may retrieve standard SLA templates from the provider."); Ex.1003, ¶¶126-127. Thus, it would have been obvious to a POSITA that the AIP's product catalog would "*receiv[e] a definition of ... a plan*" with each service added to the product catalog. Ex.1003, ¶128.

In summary, de Castro teaches registering each service application in a registry, including logging properties and status information of each application's API client. Robb teaches aggregating and publishing applications and services to a product catalog, and receiving SLA terms and criteria for the published applications and services. Their combination renders obvious "*receiving a*

definition of a service and a plan for each of the plurality of application service provider systems.” Ex.1003, ¶¶129-130.

[1.8] receiving one or more definitions of actions for the service, wherein the actions comprise responses to triggering events occurring with a telephone number; and

First, de Castro’s system includes “**a finite set of commands**” (“*one or more definitions of actions*”), where “an application program essentially issues a series of such **commands to operate the communication application server.**” Ex.1005, [0109]. De Castro teaches “providing a messaging API protocol for customers of the communication application server on the network to **execute their applications as a series of these commands.**” Ex.1005, [0118]. Using this messaging protocol (e.g., Rayo protocol), “**a command carried in a message is received**” (“*receiving*”) at the communication application server 200. Ex.1005, [0122]. That is, the de Castro’s system includes “**[r]eceiving into the application server commands issued by the API client** for invoking and manipulating the class objects relative to a call.” Ex.1005, [0135]; Ex.1003, ¶131.

At the application communication server, “the API server **parses out the command in each message for execution in turn on the associate call actor.**” Ex.1005, [0124]. The parsed “commands” define the parameters of executable actions, such as to “**Answer**” or “**Redirect**” a telephone call. Ex.1005, [0141]; Ex.1003, ¶132.

Therefore, de Castro's communication application server receives commands for a service, where the commands define telephone actions such as to answer or redirect a call, which teaches "*receiving one or more definitions of actions for the service.*" Ex.1003, ¶133.

Second, de Castro explains that a command in the "Rayo protocol primarily deals with calls" to a telephone number. Ex.1005, [0139]. For example, the command "**Redirect ... [o]nly applies to incoming calls.**" Ex.1005, [0141]. In an example, a call to the phone number "**to='tel:+18003211212'**" ("*triggering event occurring with a telephone number*") can result in a redirect command "**redirect to='tel:+14152226789'**" ("*action*"). Ex.1005, [0141]. As another example, a call to a phone number where the "**[d]ial destination is busy and cannot answer the call**" ("*triggering event occurring with a telephone number*") can result in an "**<end/> [event that] indicates that the call has been disconnected**" ("*action*"). Ex.1005, [0142]; Ex.1003, ¶134.

Therefore, the "redirect" or "disconnect" commands that execute, in response to a call to a phone number, render obvious that "*the actions comprise responses to triggering events occurring with a telephone number.*" Ex.1005, [0141]-[0142]; Ex.1003, ¶135.

Thus, de Castro teaches a communication application server receiving Rayo messages including "answer" or "redirect" commands in response to a call to a

phone number, which renders obvious “*receiving one or more definitions of actions for the service, wherein the actions comprise responses to triggering events occurring with a telephone number.*” Ex.1003, ¶135.

[1.9] publishing, after integrating each of the plurality of API interfaces with the unified services platform, the definition of the service, the definition of the plan, and the one or more definitions of actions for each application service provider in a provisioning catalog.

First, as discussed above at [1.7], de Castro teaches a registry that registers each application, including logging application “**properties and status**” information. Ex.1005, [0130]. And a POSITA would have found it obvious that such “properties and status” information for each application logged in a registry would include a name and short description of the application service. Ex.1005, [0130]; Ex.1003, ¶136.

De Castro also teaches that “API client 630 such as API client 1 630-1 which **interprets App Script 1 and renders it in terms of the commands of the finite set** defined at the API server 450.” Ex.1005, [0119]. Therefore, the application services are defined by “commands” coded in an “application script” (“*definitions of actions*”). Ex.1005, [0119]; Ex.1003, ¶137.

Robb describes that an application infrastructure platform (AIP) may “aggregate its own applications and services with those of both external ASPs/ISVs and its own ASPs into an AIP’s **Product Catalog.**” Ex.1006, 3:17-20;

Ex.1003, ¶138. “[C]ustomer ASPs may select from among numerous services in the AIP product catalog.” Ex.1006, 26:24-26.

It would have been obvious to a POSITA to apply Robb’s AIP product catalog (“*provisioning catalog*”) publishing technique to de Castro’s system. Ex.1003, ¶139. In the combination, Robb’s product catalog would publish information about de Castro’s available application scripts and associated commands (“*definitions of actions*”) for service providers to aggregate, repackage, and resell. Ex.1003, ¶139. Robb’s product catalog would also publish the related service properties (“*definition of the service*”) and default SLA terms (“*definition of the plan*”) associated with the service pack. Ex.1003, ¶139. A POSITA would have found this combination obvious at least for the reasons provided above at element [1.7] and Section XI.C.3. Ex.1003, ¶139. Thus, de Castro in view of Robb renders obvious “*publishing ... the definition of the service, the definition of the plan, and the one or more definitions of actions for each application service provider in a provisioning catalog.*” Ex.1003, ¶139.

Second, as discussed above at [1.4], de Castro teaches scripting engines that process customer framework APIs into the commands of a unified API. Ex.1005, [0109]. As a result of this “*integrating*,” clients are able to use the services hosted by the communication application server. Ex.1005, [0109]; Ex.1003, ¶140.

Therefore, it would have been obvious to a POSITA to publish the available

services to the AIP product catalog (“*publishing ... in a provisioning catalog*”) only after the services have been “*integrat[ed]*” and are available to use. Ex.1005, [0109]; Ex.1006, 34:16-17, 34:25-27; Ex.1003, ¶140.

A POSITA would have found it obvious to combine Robb’s teachings with de Castro’s system at least for the reasons provided above at element [1.7] and Section XI.C.3. Ex.1003, ¶141. In the resulting combination, a service provider-accessible product catalog would publish, for each application, a service pack including the application script (“*definitions of actions*”), application properties and status information such as the name and a short description of the service (“*definition of the service*”), and default SLA terms and conditions (“*definition of the plan*”). Ex.1003, ¶141. Indeed, it would have been obvious for the service pack to include the app script to facilitate deploying the app script to a newly instantiated scripting engine when a user activates/subscribes to the service. Ex.1003, ¶141. Therefore, a POSITA would understand that de Castro in view of Robb renders obvious performing the “*publishing*” step “*after integrating each of the plurality of API interfaces with the unified services platform.*”

Thus, de Castro teaches a registry that logs properties and status information for each application service, as well as an application script that includes executable commands. Ex.1003, ¶142. Robb teaches aggregating and publishing applications and services to a product catalog, and defining service packs having

an aggregation of services and SLA terms and conditions. In combination, these teachings render obvious “*publishing, after integrating each of the plurality of API interfaces with the unified services platform, the definition of the service, the definition of the plan, and the one or more definitions of actions for each application service provider in a provisioning catalog.*” Ex.1003, ¶142.

5. Claim 2

[2.0] The method of claim 1, further comprising: receiving, from the network provider system, a selection of one or more services from the plurality of the application services provider systems; and

First, de Castro explains that “[i]nbound calls originate... via SIP,” resulting in messaging to “**the server that’s handling the call signaling and media.**” Ex.1005, [0139]. This system “allows application clients 40 such as, for example, **VoIP phones to access the applications hosted in the communication application servers.**” Ex.1005, [0115]. For instance, a client is able to select and “**access a specific communication application** by calling a dialed number or an URL.” Ex.1005, [0116]. In this way, a particular “[communication] application server is ... **selectable for providing telephony and communication services.**” Ex.1005, claim 2, [0141]. Therefore, de Castro teaches clients on an SIP network (“*receiving, from the network provider system*”) select and access various services (“*a selection of one or more services*”). Ex.1003, ¶143.

Second, as discussed above at [1.3], de Castro teaches that an application

script for a service is hosted and “processed remotely by appropriate **scripting engines**.” (“*services from [] the application services provider systems*”). Ex.1005, [0038]; Ex.1003, ¶144.

Third, Robb explains that an administrator or profile manager can select and customize services from the AIP product catalog, and then “repackage and resell [the selected services] to their customers as corporate portals.” Ex.1006, 7:29-32. The administrator defines a publishable service package from the AIP product catalog to include “an aggregation of services for a user role.” Ex.1006, 34:15-17, 24-26. That is, an administrative user, when “adding service packs to a corporate portal,” specifies a selection of one or more services to be included in the service package. Ex.1006, 34:24-28. A POSITA would have understood that for the administrator to specify a selection of services to include in the service package, de Castro’s system would “*receiv[e]*” the selection of services from the product catalog specifying the service package to be added to the corporate portal. Ex.1003, ¶145. Robb describes that this may be performed by a profile manager, administrator, etc., who creates and updates the corporate portal; it would have been obvious that these selections can be received by de Castro’s system from a manager or administrator connected via the SIP network (“*receiving, from the network provider system, a selection*”). Ex.1006, 34:16-26; Ex.1003, ¶145.

Thus, de Castro and Robb render obvious “*receiving, from the network*

provider system, a selection of one or more services from the plurality of the application services provider systems.” Ex.1003, ¶146.

[2.1] publishing the selection of the one or more services in a services catalog,

Robb’s “product catalog” teaches the “***provisioning catalog***” of [1.9]. *Supra*, Section XI.C.4.[1.9]. Here, [2.1] recites “*publishing ... in a services catalog.*” As signaled above at Section XI.C.3 and further explained in this section below, Robb’s “corporate portal” teaches the “*services catalog.*” Ex.1003, ¶147.

Robb describes that the AIP “aggregate[s] its own applications and services with those of both external ASPs/Internet Service Vendors (ISVs) and its own ASPs into a ‘Product Catalog,’ which “may then be subscribed to by its ASPs.” Ex.1006, 7:53-57. The ASPs may select services from the catalog and “repackage and resell [the selected services] to their customers as corporate portals.” Ex.1006, 7:29-32. Thus, Robb teaches that the AIP product catalog (“*provisioning catalog*”) is used by ASPs to select and customize service packs, which they provide to their customers in their customized corporate portal (“*services catalog*”). Ex.1003, ¶148. Repackaging and offering service packs for resale to customers through a corporate portal teaches “*publishing ... in a services catalog.*” Ex.1003, ¶148.

It would have been obvious to a POSITA to apply Robb’s corporate portal publishing technique to de Castro’s system. Ex.1005, [0130], [0141]; Ex.1006, 34:16-17, 34:25-27; Ex.1003, ¶149. In the combination, de Castro’s system would

aggregate available services into a service-provider-facing product catalog. The service providers would then publish services selected and customized by the service providers included in a user-facing corporate portal (“*publishing the selection ... in a services catalog*”). Ex.1006, 34:16-26; Ex.1003, ¶149.

Thus, de Castro teaches application services that are selectable and accessed by a call from an SIP network client. Robb further teaches publishing a service pack of selected services to a portal, which in combination with de Castro’s teachings renders obvious “*publishing the selection of the one or more services in a services catalog.*” Ex.1003, ¶150.

[2.2] *wherein the services catalog is accessible by one or more subscribers of the network provider system.*

As discussed above at [2.1], it would have been obvious to a POSITA to apply Robb’s portal publishing technique (“*services catalog is accessible*”), to de Castro’s system, to publish in a portal the services available to clients on a SIP network (“... *by one or more subscribers of the network provider system*”). Ex.1005, [0130], [0139], [0141]; Ex.1006, 34:16-17, 34:25-27; Ex.1003, ¶151. As an example, Robb describes a “**adding service packs to a corporate portal,**” where a service package includes “an aggregation of services for a user role.” Ex.1006, 34:16-17. That is, Robb describes “**offer[ing] ... services to the employees of corporations through the corporate portal.**” Ex.1006, 6:29-30. In

this way, “[t]hese corporate web sites (or corporate portals) provide the employees with a wide variety of services.” Ex.1006, 9:5-7; Ex.1003, ¶151.

Thus, de Castro teaches application services that are selectable and accessed by a call from an SIP network client, and Robb teaches publishing a portal of services accessible to SIP clients, which renders obvious “*wherein the services catalog is accessible by one or more subscribers of the network provider system.*” Ex.1003, ¶152.

6. Claim 3

[3.0] *The method of claim 1, further comprising: receiving a request for associating at least one service from the plurality of application service provider systems with a telephone number of a subscriber;*

First, de Castro explains that “application clients such as VoIP phones 30, [are] able to **access a specific communication application by calling a dialed number** or an URL. The call is routed ... to one of the communication application servers 200[.]” Ex.1005, [0116]. A POSITA would have understood that the application server receiving the client’s call to provide access to a communication application teaches “*receiving a request for associating [a] service ... with ... a subscriber.*” Ex.1005, [0116]; Ex.1003, ¶153.

Therefore, de Castro teaches one of the application servers 200 receiving, from an application client (“*subscriber*”), a call to access to one of the communication applications, which renders obvious “*receiving a request for*

associating [a] service ... with ... a subscriber.” Ex.1003, ¶154.

Second, de Castro describes these requests for services as controlling “Incoming Calls” and Outbound Calls” made to and from the phone numbers of clients. Ex.1005, [0138]-[0142]. As discussed above at [1.2], de Castro gives an example Rayo protocol message including code illustrating that the service is associated with the “from” telephone number of “*a subscriber*” via the call’s JID:

<!-- Message comes from the Call’s JID -->

...

<offer xmlns='urn:xmpp:Rayo:1'

to='tel:+18003211212'

from='tel:+13058881212'>

Ex.1005, [0141]; Ex.1003, ¶155.

Additionally, de Castro describes an exemplary “Follow me, find me” service where a user is associated with multiple telephone numbers. Ex.1005, [0007]. It would have been obvious to a POSITA that when a user configures such a service, de Castro’s systems would receive a request to associate the service with the user’s telephone numbers. Ex.1003, ¶156. These teachings from de Castro render obvious “*a request for associating [a] service ... with a telephone number of a subscriber.*” Ex.1003, ¶156.

Thus, de Castro teaches application servers 200 controlling calls to and from an application client, who is associated with a phone number(s), to access a

communication application, such as an inbound or outbound call control application, which renders obvious “*receiving a request for associating at least one service from the plurality of application service provider systems with a telephone number of a subscriber.*” Ex.1003, ¶157.

[3.1] associating the telephone number with the at least one service; and

First, de Castro teaches that, responsive to a subscriber’s inbound call requesting to use a service, the call is offered to a service provider by associating the telephone number with a Jabber Identifier or JID. Ex.1005, [0139]-[0141]; Ex.1003, ¶158.

De Castro discloses that “inbound calls ... are offered to Rayo clients via XMPP using a Jabber Identifier (JID)” —such that the JID associates the call with a Rayo client (i.e., app script in a scripting engine) that controls the call as well as the SIP client who placed the call. Ex.1005, [0139]; Ex.1003, ¶159. As an example, de Castro describes identifying an inbound call using both a JID and a “from” telephone number:

```
<!-- Message comes from the Call’s JID -->
<presence to=‘16577@app.Rayo.net’
from=‘9f00061@call.Rayo.net/1’
  <offer xmlns=‘urn:xmpp:Rayo:1’
    to=‘tel:+18003211212’
    from=‘tel:+13058881212’>
```

...

Ex.1005, [0141]; Ex.1003, ¶160.

De Castro further teaches using the Jabber ID (JID) (which is mapped to a subscriber's "*telephone number*") to execute a particular communication

application ("*associating the telephone number with the at least one service*").

Ex.1005, [0139]. For instance, the service provided by the scripting engine or

"Rayo client can now control the call" by issuing commands that identify the call

using the JID. Ex.1005, [0141]. De Castro also states that "[e]ach call is in turn

represented by its own unique JID allowing a two way conversation between

the Rayo client and the server that's handling the call signaling and media."

Ex.1005, [0139]; Ex.1003, ¶161.

Therefore, de Castro teaches that the JID is mapped to the telephone number ("*the telephone number*") of the client accessing the communication application;

and the communication application server 200 associates the telephone number

with the JID, which is associated with the communication application. Further, the

service, which may be accessed by a phone call, is associated with the phone

number because the server is maintaining the call with the client. Ex.1005, [0139];

Ex.1003, ¶162.

Second, alternatively, de Castro renders this element obvious in teaching that a client is able to "**access a specific communication application by calling a**

dialed number.” Ex.1005, [0102]. That is, an application developer (“*subscriber*”) can associate their phone number with the service they developed and host on a scripting engine, by allowing other clients to access the service when the phone number is dialed. Ex.1005, [0102]. Therefore, a POSITA would have found that making a service accessible by dialing a telephone number renders obvious “*associating the telephone number with the at least one service.*” Ex.1003, ¶163.

Third, alternatively, the analysis at [3.0] shows the obviousness of de Castro’s systems receiving a request to associate a user’s telephone number(s) with a service. It would have been obvious to a POSITA for de Castro’s systems to respond by performing the requested association. Ex.1003, ¶164.

Thus, de Castro renders obvious in multiple ways “*associating the telephone number with the at least one service.*” Ex.1003, ¶165.

[3.2] invoking the at least one service in response to a communication event with the telephone number.

It would have been obvious to a POSITA that de Castro—which teaches a service that processes call control commands and routes the service to a subscriber’s phone number—renders obvious “*invoking the at least one service in response to a communication event with the telephone number.*” Ex.1005, [0116], [0141]; Ex.1003, ¶166.

De Castro teaches, for example, routing calls to a communication

application, which executes call control commands, “by looking up a directory service DIR0 16 with the IP address/dialed number in order to return the address of an available application platform with the necessary resources to execute the specific communication application.” Ex.1005, [0102]. It would have been obvious that, for the incoming call either to or from the client’s 40 telephone number (“*a communication event with the telephone number*”), the directory DIR0 routes the call to/from the client and the server that executes the communication application (“*invoking the [] service*”); the server “can now control the call by using ... commands,” and the actions of the commands can be routed to the client 40. Ex.1005, [0141]; *see also id.* [0007] (discussing activation of various services, such as “Follow me, find me,” in response to inbound calls); Ex.1003, ¶¶166-167.

Thus, de Castro teaches routing commands to a communication application server for execution, in response to a call from a client’s telephone number, which renders obvious “*invoking the at least one service in response to a communication event with the telephone number.*” Ex.1003, ¶168.

7. Claim 5

[5.0] *The method of claim 3, wherein invoking the at least one service in response to the communication event with the telephone number comprises: invoking the at least one service associated with the telephone number in response to a first triggering event, wherein the first service is provided by a first service provided by [sic] a first application service provider system of the plurality of application service provider systems.*

First, as discussed above at [3.2], de Castro teaches routing commands to a communication application server for execution in response to an inbound call (“*triggering event*”) from a client’s telephone number (“*invoking the at least one service associated with the telephone number in response to a first triggering event*”). Ex.1005, [0039], [0116], [0138]-[0142]; Ex.1003, ¶169.

Second, as discussed above at [1.2], de Castro teaches a scripting engine (“*a first application service provider system*”) processing Rayo messages to provide a service (“*first service*”), such as controlling calls made to or from the phone numbers of telephone-service clients (“*wherein the first service is provided by a first service provided by [sic] a first application service provider system of the plurality of application service provider systems*”). Ex.1005, [0116], [0141]; Ex.1003, ¶170.

Thus, de Castro teaches routing commands to a communication application server for execution using a JID, in response to a call from a client’s telephone number associated with the JID. Ex.1003, ¶171. Further, a scripting engine provides a service hosted at the communication application server, which renders obvious “*wherein invoking the at least one service in response to the communication event with the telephone number comprises: invoking the at least one service associated with the telephone number in response to a first triggering event, wherein the first service is provided by a first service provided by [sic] a*

first application service provider system of the plurality of application service provider systems.” Ex.1003, ¶171.

8. Claim 6

[6.0] *The method of claim 5, wherein the triggering event comprises a representational state transfer (REST) upon sending a message, a REST upon receiving a message, a REST before making a call, a REST after making a call, a REST before receiving a call, a REST after receiving a call, a Session Initiation Protocols (SIP) on receiving a call, a SIP on making a call, or any combination thereof.*

As discussed above at [5.0], de Castro teaches routing commands to a communication application server for execution in response to an inbound call (“*triggering event*”) from a client’s telephone number. Ex.1005, [0116], [0141]; Ex.1003, ¶172.

De Castro further describes the inbound call as using Session Initiation Protocols (SIP) (“*the triggering event comprises ... a Session Initiation Protocols (SIP) on receiving a call*”). Ex.1005, [0141]. De Castro explains that “[i]nbound calls originate from the PSTN or **via SIP.**” Ex.1005, [0139]. In an example Rayo protocol message for an Incoming Call service, the message includes code illustrating that receiving the incoming call includes using headers from the SIP protocol:

```
<!-- Signaling (e.g. SIP) Headers -->
<header name='Via' value='192.168.0.1' />
<header name='Contact' value='192.168.0.1' />
```

...

<!-- Accept (e.g. SIP 180/Ringing). Only applies to incoming calls. -->

Ex.1005, [0141]; Ex.1003, ¶173.

De Castro further explains that “the call control API is the implementation of a standards-based SIP servlet container 340 and a **SIP servlet call control API 350**”—such that receiving a call includes using SIP for call control. Ex.1005, [0117], [0141]; Ex.1003, ¶174.

Thus, de Castro teaches receiving an incoming call including using SIP protocol to control receiving a call, which renders obvious “*wherein the triggering event comprises ... a Session Initiation Protocols (SIP) on receiving a call.*” Ex.1003, ¶175.

XII. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION

A. No basis for 35 U.S.C. § 325(d) denial

Neither de Castro nor Robb were cited or considered by the Examiner during prosecution of the ’607 patent.

Accordingly, discretionary denial under 35 U.S.C. § 325(d) is not appropriate because the first part of the *Advanced Bionics* framework is not met. *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 at 8 (PTAB Feb. 13, 2020) (precedential).

B. The *Fintiv* Factors Favor Institution under § 314

The six factors for considering denial under § 314 strongly favor institution. *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential). The district court case is in its early stages. Patent Owner filed its complaint on May 20, 2022 (Ex.1020) and served Cisco on May 27, 2022. Patent Owner provided its preliminary infringement contentions on September 22, 2022. Ex.1019. Petitioner has since worked diligently to prepare this Petition, which is being filed well within the one-year timeframe allowed by Congress under 35 U.S.C. § 315(b).

1. Whether evidence exists that a stay may be granted if a proceeding is instituted

A motion to stay has not yet been filed and the Board should not infer or attempt to predict the outcome if such a motion is later filed. *Sand Revolution II LLC v. Continental Intermodal Group – Trucking LLC*, IPR2019-01393, Paper 24 at 7 (June 16, 2020) (informative); *see also Dish Network L.L.C. v. Broadband iTV, Inc.*, IPR2020-01359, Paper 15 at 11-12 (Feb. 12, 2021) (“It would be improper to speculate, at this stage, what the Texas court might do regarding a motion to stay...”). Thus, this factor is neutral.

2. Proximity of the court’s trial date to the board’s projected statutory deadline for a final written decision

The operative Docket Control Order sets July 13, 2023, for the *Markman* hearing and January 22, 2024, for Jury Selection. Ex.1018, p. 1, 3. The Final Written Decision is expected in this case in September 2024. This factor nevertheless is neutral, because the Petitioner has worked expeditiously to prepare and file this petition within six months of receiving Patent Owner’s infringement contentions on September 22, 2022. Ex.1019; *Fintiv*, Paper 11 at 11-12.

The current average time-to-trial for the district court hearing the co-pending case is 19 months. Ex.1017, p. 35; *In re Apple Inc.*, 979 F.3d 1332, 1350 (Fed. Cir. 2020). Thus, a trial in the co-pending case would be expected in December 2023. However, “scheduled trial dates are often subject to change.” *In re Apple Inc.*, 979 F.3d 1332, 1344, n.5 (Fed. Cir. 2020). Petitioner notes that four other cases are scheduled for Jury Selection on January 22, 2024—the same Jury Selection date as the co-pending litigation—and have not settled. Ex.1028 (listing all cases having Jury Selection scheduled for January 22, 2024).

As such, this factor is neutral. *See Sand Revolution II, LLC v. Continental Intermodal Grp.—Trucking LLC*, IPR2020-01393, Paper 24 (June 16, 2020) (panel granting institution on rehearing based on finding that uncertainty over district court’s trial date weighed against discretionary denial).

3. Investment in the parallel proceeding

The co-pending litigation is in its early stages, and the investment in it has been minimal. Fact discovery is not set to close until September 5, 2023, and expert discovery is not set to close until October 10, 2023. Ex.1018. The Docket Control Order sets July 13, 2023 as the date for the *Markman* Hearing. Ex.1018, p. 3. Thus, this factor weights against discretionary denial. *See PEAG LLC v. Varta Microbattery GMBH*, IPR2020-01214, Paper 8 at 17 (Jan. 6, 2021) (finding that since no claim construction hearing had yet been held and discovery was not completed, the little investment in the parallel proceeding weighed against discretionary denial).

4. Overlap between issues raised in the petition and in the parallel proceeding

There will be no overlap of prior art issues. Petitioner stipulates that, if the Board institutes trial, Petitioner will cease asserting in the co-pending litigation the combination of references on which trial is instituted for the claims, to the extent Petitioner even asserts the same combination in the district court. This factor weighs against discretionary denial. *See Verizon Bus. Network Servs. v. Huawei Techs. Co., Ltd.*, IPR2020-01079, Paper 10 at 38 (Jan. 14, 2021) (finding a similar stipulation “mitigates the concern about overlapping issues” and weighs “against discretionary denial of the Petition.”).

5. Whether the petitioner is a defendant

Petitioner is a defendant in the litigation. Ex.1020. This is true of most Petitioners in IPR proceedings. While the *Fintiv* case indicates that a difference between the district court defendant and the petitioner may weigh against discretionary denial, nothing within the *Fintiv* case suggests that the same party between the proceedings weighs in favor of discretionary denial. *See HP Inc. v. Slingshot Printing LLC*, IPR2020-01084, Paper 13 at 9 (Jan. 14, 2021) (having the “same parties as parallel proceeding” makes factor 5 “neutral”). Accordingly, this factor is neutral and should not be a basis for denying institution.

6. Other circumstances that impact the Board’s exercise of discretion, including the merits

As discussed above, the prior art presented in this Petition renders the Challenged Claims unpatentable as obvious. In view of the striking similarities between the ’607 patent claims and both de Castro and Robb, this petition “presents compelling evidence of unpatentability.” Director Vidal’s Guidance Memorandum, Ex.1021, 2. This factor weighs against discretionary denial.

As such, the *Fintiv* factors are either neutral or weigh against discretionary denial. Because this Petition was filed two months before the statutory bar date, institution should not be denied on discretionary factors.

C. Discretionary denial under *General Plastic* is not appropriate

The '607 patent has not been challenged in any prior IPR petition, so no *General Plastic* discretionary institution factors apply here. *See General Plastic Indus. Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 16 (Sept. 6, 2017) (Section II.B.4.i. precedential).

XIII. CONCLUSION

Petitioner has established a reasonable likelihood that the Challenged Claims are unpatentable.

Respectfully submitted,

Dated: March 21, 2023
HAYNES AND BOONE, LLP
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Customer No. 27683

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

XIV. MANDATORY NOTICES

A. Real party-in-interest

Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner certifies that the real party-in-interest is Cisco Systems, Inc.

B. Related matters

Pursuant to 37 C.F.R. § 42.8(b)(2), to the best knowledge of the Petitioner, the '607 patent is or was involved in the following cases:

Case Heading	Number	Court	Filed
<i>Zilkr Cloud Technologies, LLC v. Cisco Systems, Inc.</i>	2-22-cv-00166	EDTX	May 20, 2022
<i>Zilkr Cloud Technologies, LLC v. RingCentral, Inc.</i>	3-21-cv-02807	NDTX	Nov. 11, 2021

Additionally, the '607 patent is related to U.S. Pat. No. 9,210,254, which is involved in the *Inter Partes* Review proceeding IPR2023-00553.

C. Lead and back-up counsel and service information

Lead Counsel

Theodore M. Foster
HAYNES AND BOONE, LLP
2323 Victory Ave. Suite 700
Dallas, TX 75219

Phone: (303) 382-6205
Fax: (214) 200-0853
ipr.theo.foster@haynesboone.com
USPTO Reg. No. 57,456

Back-up Counsel

David L. McCombs
HAYNES AND BOONE, LLP
2323 Victory Ave. Suite 700
Dallas, TX 75219

Phone: 214-651-5533
Fax: 214-200-0853
david.mccombs.ipr@haynesboone.com
USPTO Reg. No. 32,271

Eugene Goryunov
HAYNES AND BOONE, LLP
2323 Victory Ave. Suite 700
Dallas, TX 75219

Phone: (312) 216-1630
Fax: (214) 200-0853
eugene.goryunov.ipr@haynesboone.com
USPTO Reg. No. 61,579

Samuel Drezdson
HAYNES AND BOONE, LLP
2323 Victory Ave. Suite 700
Dallas, TX 75219

Phone: (972) 739-6918
Fax: (214) 200-0853
samuel.drezdson.ipr@haynesboone.com
USPTO Reg. No. 67,085

Brett Bostrom
HAYNES AND BOONE, LLP
2323 Victory Ave. Suite 700
Dallas, TX 75219

Phone: (971)739-6973
Fax: (214) 200-0853
brett.bostrom.ipr@haynesboone.com
USPTO Reg. No. 73,009

Please address all correspondence in this proceeding to lead and back-up counsel. Petitioner consents to service in this proceeding by email at the addresses above.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner hereby certifies, in accordance with and reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 11,831. Pursuant to 37 C.F.R. § 42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: March 21, 2023

/Theodore M. Foster/

Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

CERTIFICATE OF SERVICE

The undersigned certifies that, in accordance with 37 C.F.R. § 42.6(e) and 37 C.F.R. § 42.105, service was made on Patent Owner as detailed below.

Date of service March 21, 2023

Manner of service FEDERAL EXPRESS

Documents served Petition for *Inter Partes* Review Under 35 U.S.C. § 312
and 37 C.F.R. § 42.104 of U.S. 9,998,607;
Petitioner's Exhibit List;
Exhibits 1001-1028

Persons served Conley Rose, P.C.
4965 Preston Park Blvd, Suite 195E
Plano, TX 75093

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456