US 20230351403A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2023/0351403 A1**

Johnson et al. (43) **Pub. Date: Nov. 2, 2023**

(54) **PROGRAMMABLE TIMELINE FEATURE FOR COMMUNICATION SYSTEMS**

(71) Applicant: **Twilio Inc.**, San Francisco, CA (US)

(72) Inventors: **Gavin Johnson**, San Francisco, CA (US); **Amool Gupta**, Berkely, CA (US); **Charlie Helin**, Murrieta, CA (US)

(21) Appl. No.: **18/198,527**

(22) Filed: **May 17, 2023**

Related U.S. Application Data

(60) Provisional application No. 63/363,935, filed on Apr. 29, 2022.

Publication Classification

(51) **Int. Cl.**

G06Q 30/016 (2006.01)

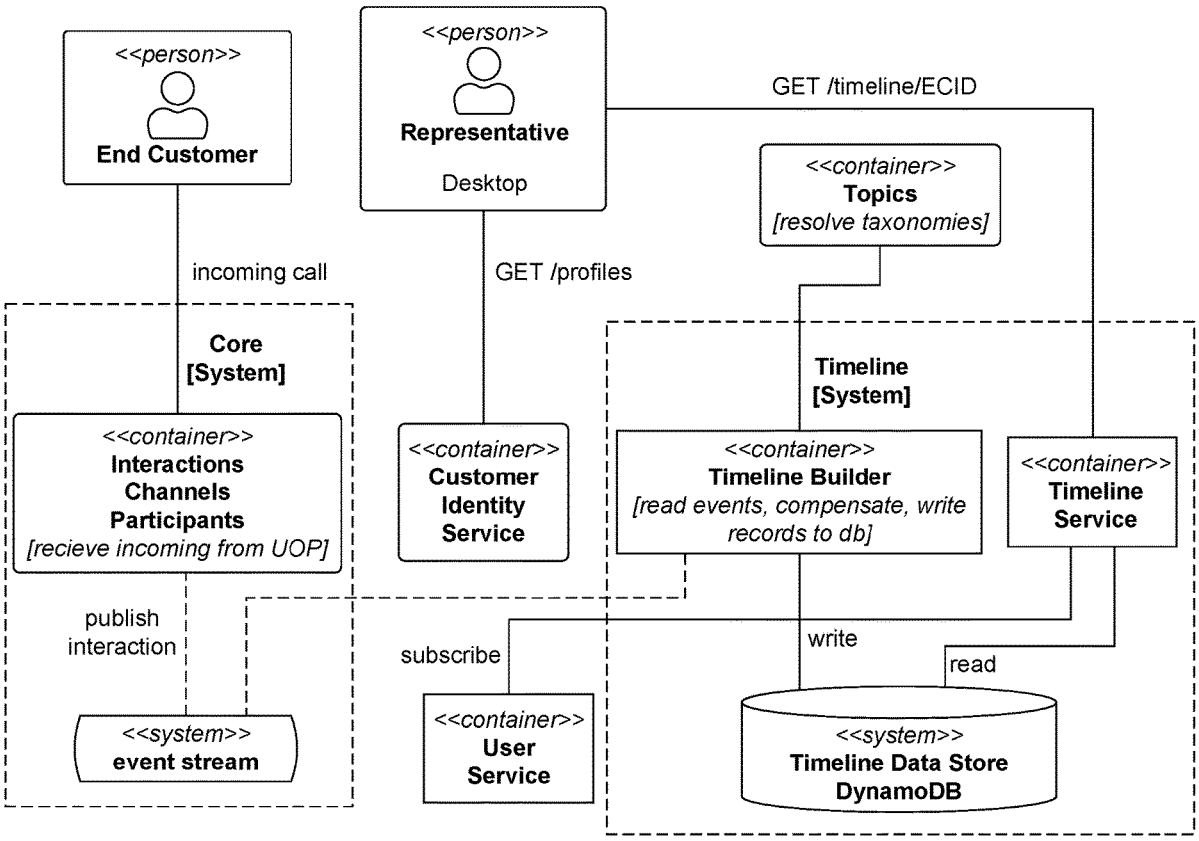
G06F 9/451 (2006.01)

(52) **U.S. Cl.**

CPC ***G06Q 30/016*** (2013.01); ***G06F 9/451*** (2018.02)

(57) **ABSTRACT**

A method of processing one or more interactions using a timeline is disclosed. Data pertaining to a plurality of omnichannel interactions is stored in a timeline record in real-time when each of the plurality of interactions occurs. Each of the plurality of interactions is associated with one or more topics and an identifier of an end customer. A request for a subset of the data that corresponds to an additional interaction of the end customer is received via an API. The subset identified in one or more parameters of the API. In response to the request, the subset of the data is provided for surfacing in a graphical user interface such that one or more representatives can make a determination of whether to accept an invitation to process the additional interaction.



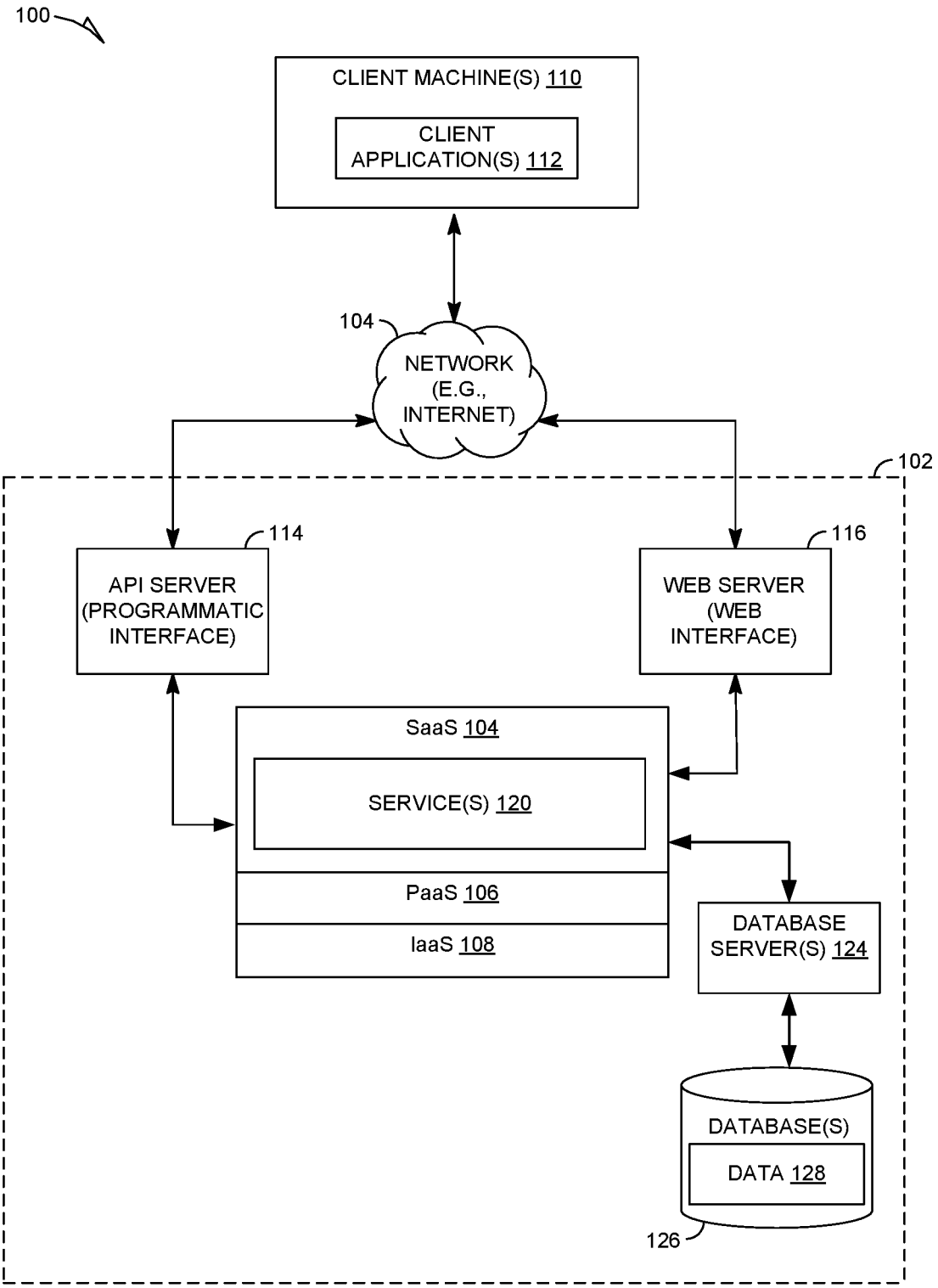

100
CLIENT MACHINE(S) 110
CLIENT APPLICATION(S) 112
104
NETWORK (E.G., INTERNET)
102
114
116
API SERVER (PROGRAMMATIC INTERFACE)
WEB SERVER (WEB INTERFACE)
SaaS 104
SERVICE(S) 120
PaaS 106
IaaS 108
DATABASE SERVER(S) 124
DATABASE(S)
DATA 128
126

FIG. 1

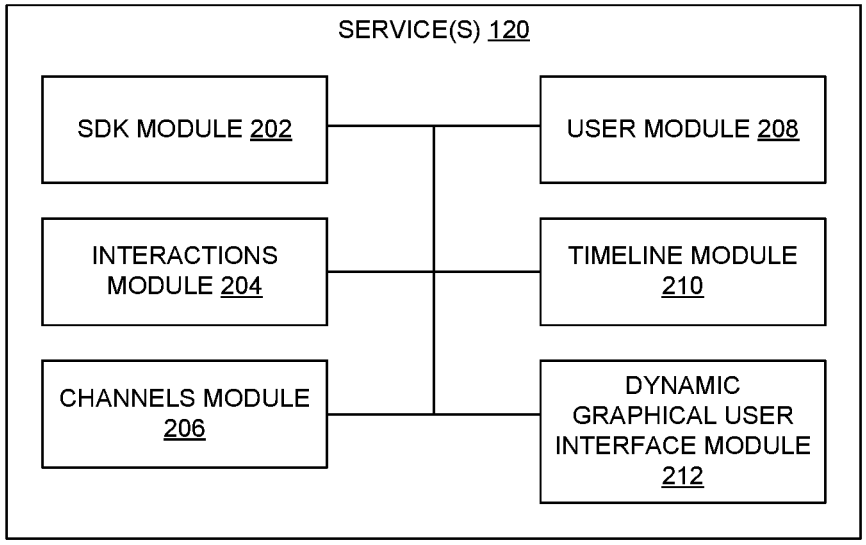

SERVICE(S) 120
SDK MODULE 202
USER MODULE 208
INTERACTIONS MODULE 204
TIMELINE MODULE 210
CHANNELS MODULE 206
DYNAMIC GRAPHICAL USER INTERFACE MODULE 212

FIG. 2

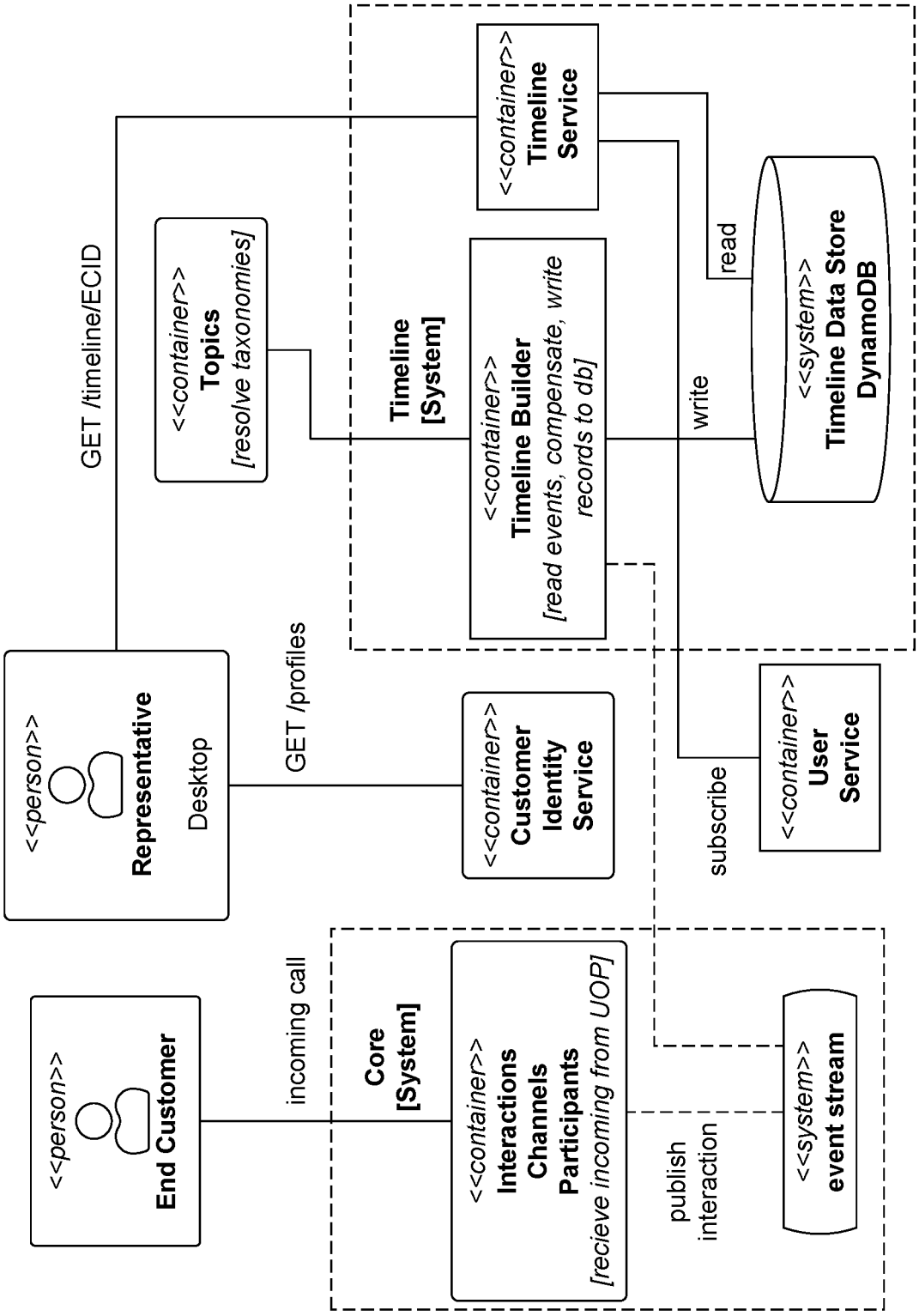

<<person>>
End Customer
incoming call
Core
[System]
<<container>>
Interactions
Channels
Participants
[recieve incoming from UOP]
publish
interaction
<<system>>
event stream
<<person>>
Representative
Desktop
GET /profiles
<<container>>
Customer
Identity
Service
GET /timeline/ECID
<<container>>
Topics
[resolve taxonomies]
Timeline
[System]
<<container>>
Timeline Builder
[read events, compensate, write
records to db]
<<container>>
Timeline
Service
write
read
<<system>>
Timeline Data Store
DynamoDB
subscribe
<<container>>
User
Service

FIG. 3

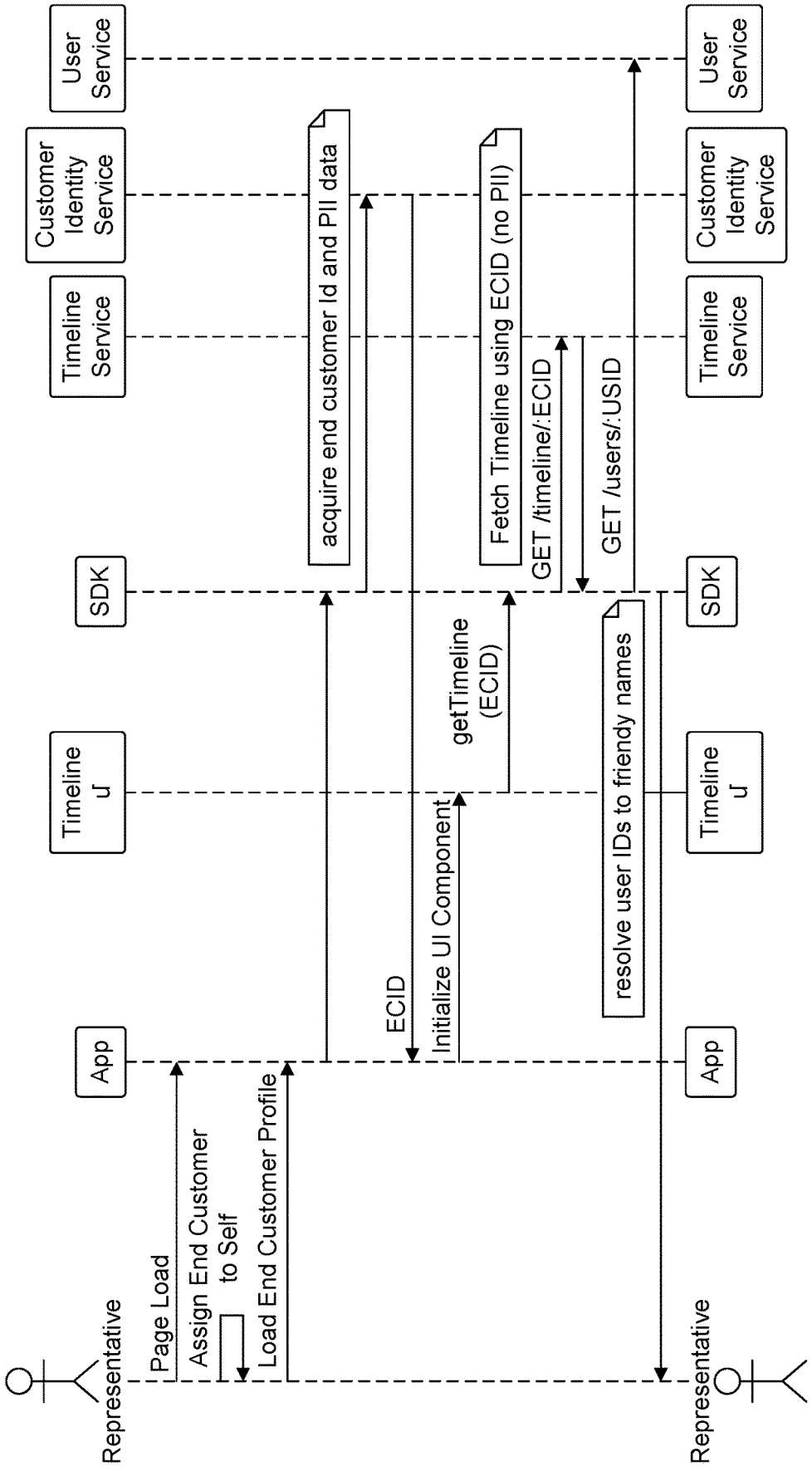

Representative
Page Load
Assign End Customer to Self
Load End Customer Profile
App
Timeline UI
SDK
Timeline Service
Customer Identity Service
User Service
ECID
Initialize UI Component
getTimeline (ECID)
acquire end customer Id and PII data
GET /timeline/:ECID
Fetch Timeline using ECID (no PII)
GET /users/:USID
resolve user IDs to friendy names
App
Timeline UI
SDK
Timeline Service
Customer Identity Service
User Service
Representative

FIG. 4

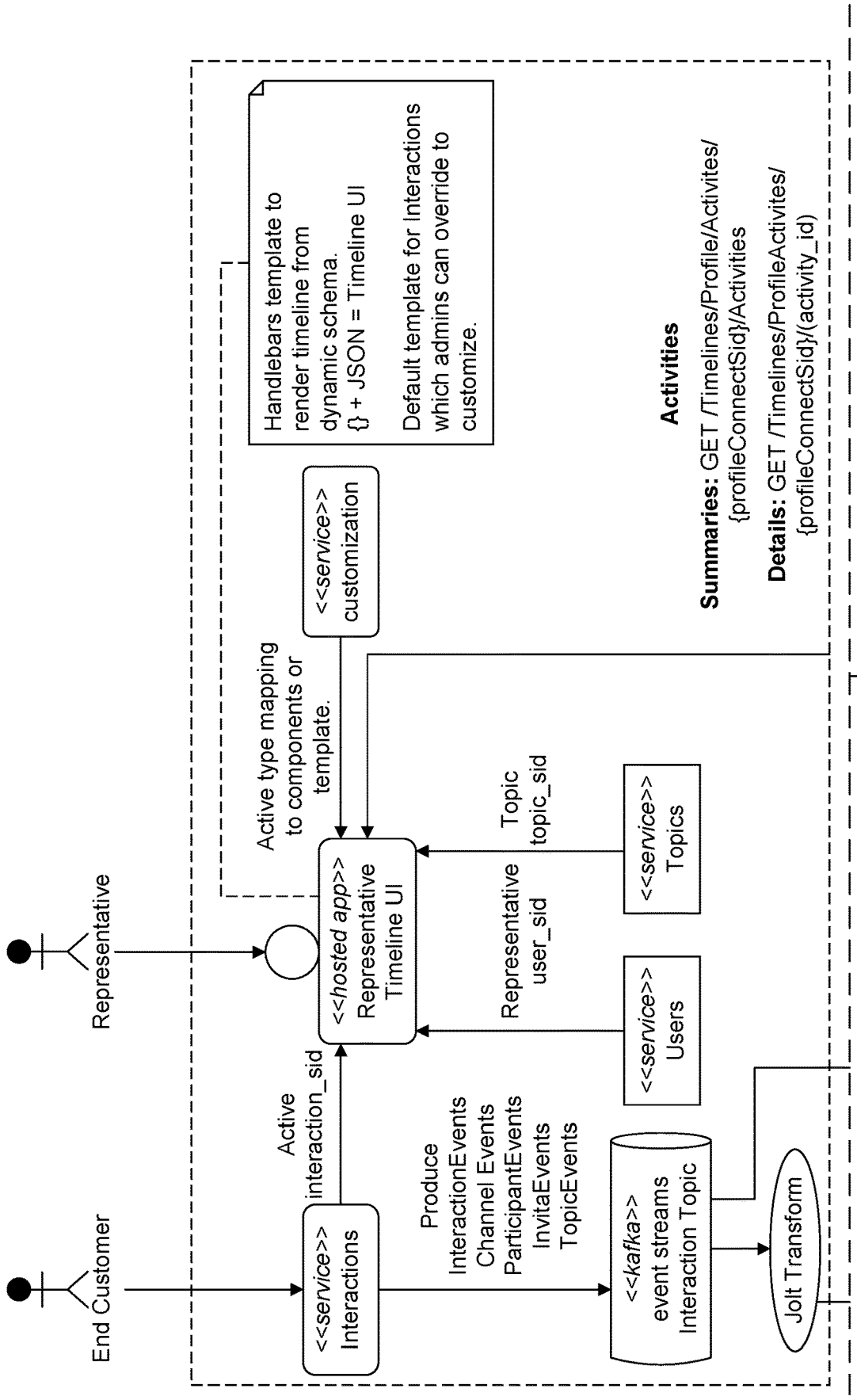

End Customer
Representative
<<service>>
Interactions
Active
interaction_sid
<<hosted app>>
Representative
Timeline UI
Active type mapping
to components or
template.
<<service>>
customization
Handlebars template to
render timeline from
dynamic schema.
{} + JSON = Timeline UI
Default template for Interactions
which admins can override to
customize.
Activities
Summaries: GET /Timelines/Profile/Activites/
{profileConnectSid}/Activities
Details: GET /Timelines/ProfileActivites/
{profileConnectSid}/(activity_id)
Produce
InteractionEvents
Channel Events
ParticipantEvents
InvitaEvents
TopicEvents
<<kafka>>
event streams
Interaction Topic
Jolt Transform
Representative
user_sid
<<service>>
Users
Topic
topic_sid
<<service>>
Topics
A

FIG. 5

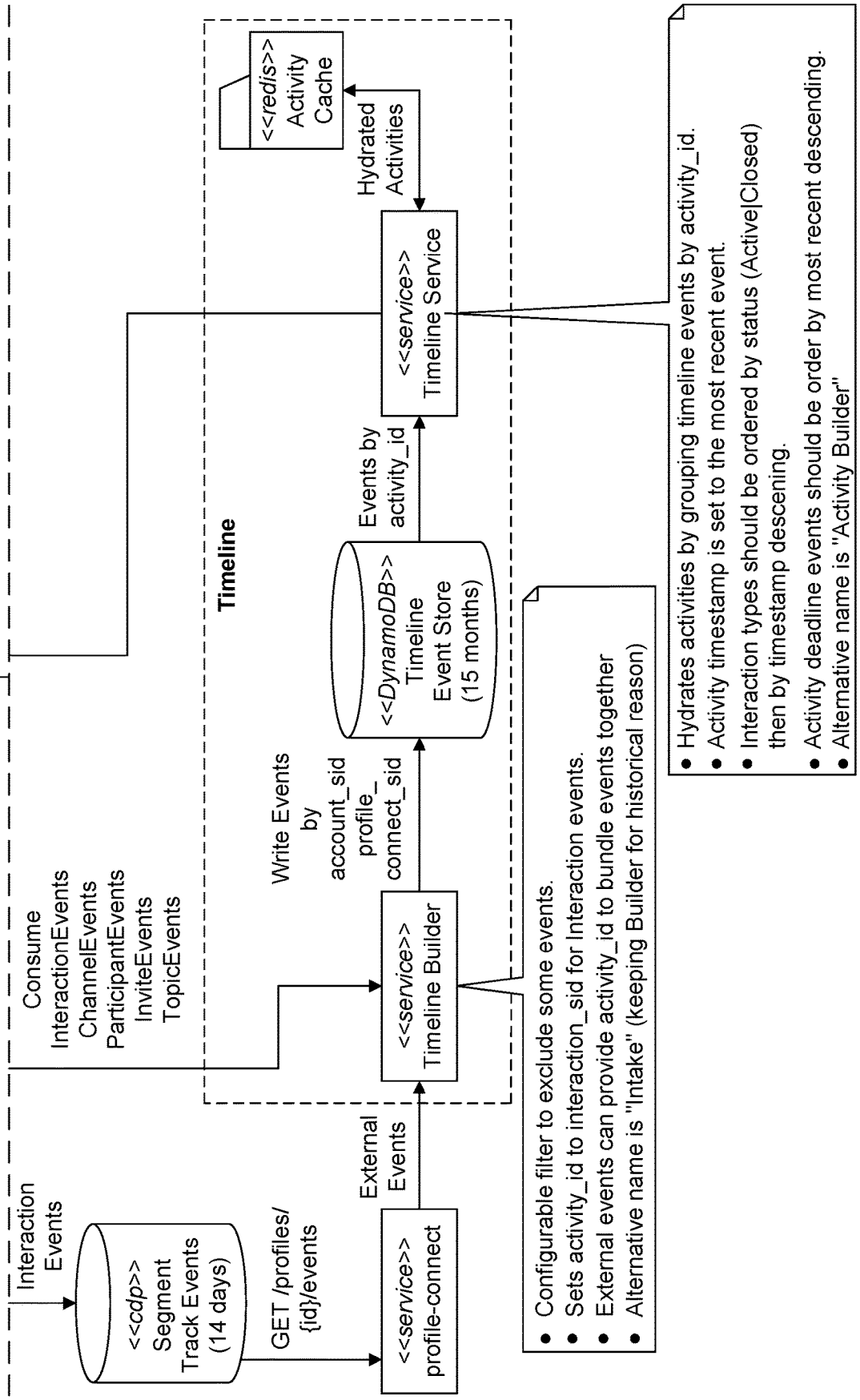


FIG. 5(Continued)

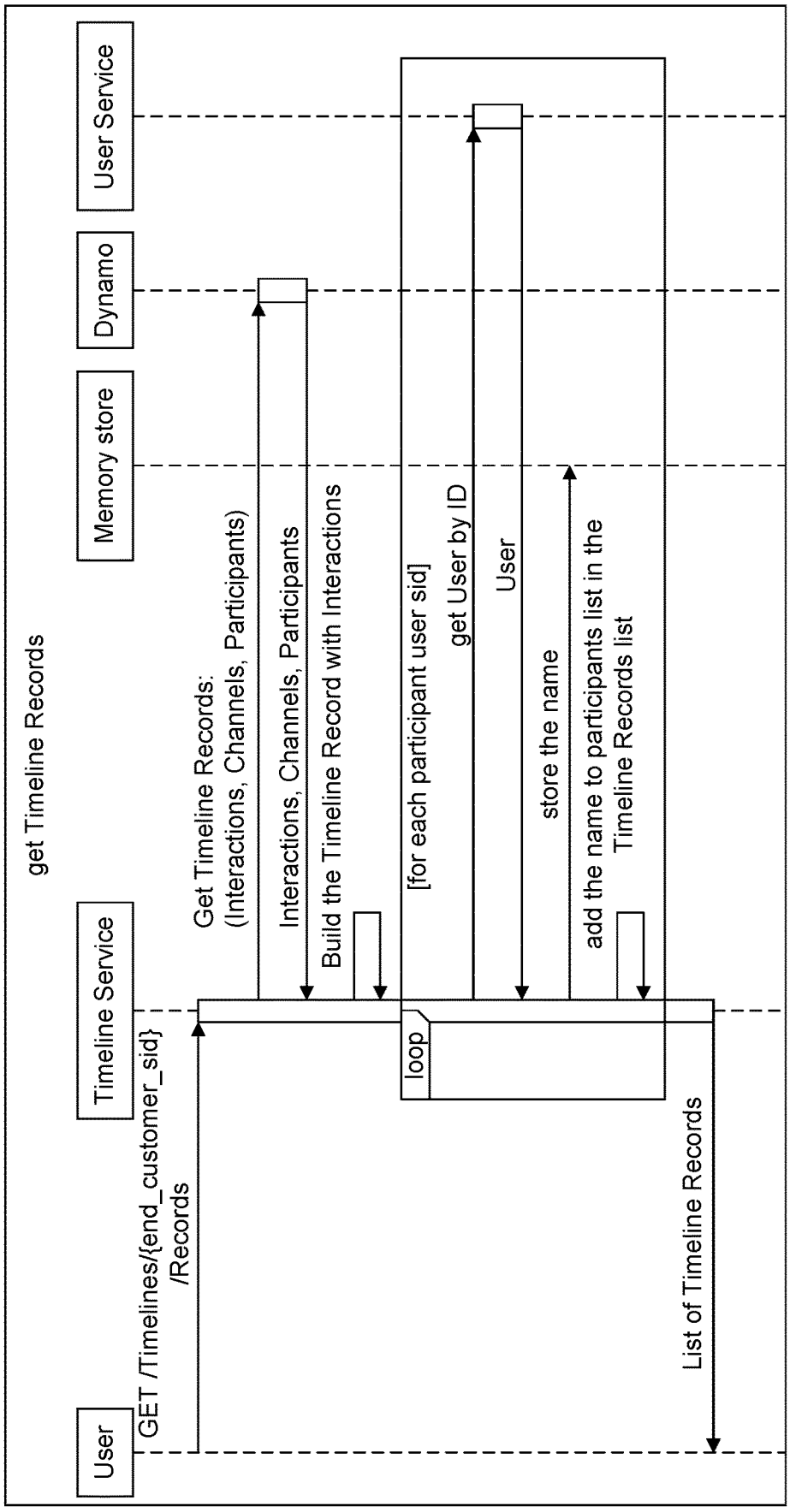

get Timeline Records
User
Timeline Service
Memory store
Dynamo
User Service
GET /Timelines/{end_customer_sid}
/Records
Get Timeline Records:
(Interactions, Channels, Participants)
Interactions, Channels, Participants
Build the Timeline Record with Interactions
loop
[for each participant user sid]
get User by ID
User
store the name
add the name to participants list in the
Timeline Records list
List of Timeline Records

FIG. 6

```
[
    {
      "sid": "ZZ00000000000000000000000000000001",
      "flex_instance_sid": GO00000000000000000000000000000001",
      "customer_sid": "YI00000000000000000000000000000001",
      "version" : 1,
      "date_created": "2021-01-30T08:30:00000Z",
      "date_updated": "2021-01-30T08:30:00000Z",
      "data": {
        "sid": "KD00000000000000000000000000000001",
        "data_type": "INTERACTION" ,
```


FIG. 7A

```
        "status": "ACTIVE",
        "version": 1,
        "date_created": "2021-01-30T08:30:00000Z",
        "date_modified": "2021-01-30T08:30:00000Z",
        "topics": [
          {
       "sid": "ZL00000000000000000000000000000001"
          }
        ],
        "channels": [
          {
            "sid": "KE00000000000000000000000000000001"
            "version": 1,
            "type": "SMS",
            "status": "ACTIVE",
            "initiated_by": "CUSTOMER",
            "date_created": "2021-01-30T08:30:00000Z",
            "date_modified": 2021-01-30T08:30:00000Z",
            "participants": [
              {
                "sid": "KJ00000000000000000000000000000001"
                "user_sid": "US00000000000000000000000000000001",
                "type": "CUSTOMER",
                "status": "JOINED",
                "version": 1,
                "date_created": "2021-01-30T08:30:00000Z",
                "date_modified": 2021-01-30T08:30:00000Z",
              }
            ]
          }
        ]
      }
    },
{
  "sid": "ZZ...1",
  "flex_instance_sid": "GO...1",
  "version":"1",
  "customer_sid":"YI...1",
  "date_created": "2021-01-30T08:30:00000Z",
  "date_updated": 2021-01-30T08:30:00000Z",
  "data":{
      "data_type":"EXTERNAL",//SEGMENT//OTHER_EVENT_SOURCES
      "sid":"KD...1",
```


FIG. 7B

```
            "version":"1.0",
            // External events to be defined after private beta,
            "date_created":"2021-01-31T09:00:00Z",
            "date_modified":"2021-02-07T10:37:00Z",
        }
    ],
    "meta": {
        "previous_token": "string",
        "next_token": "string",
        "list_key": "string",
        "direct_token": true
    }
]
```


FIG. 7C

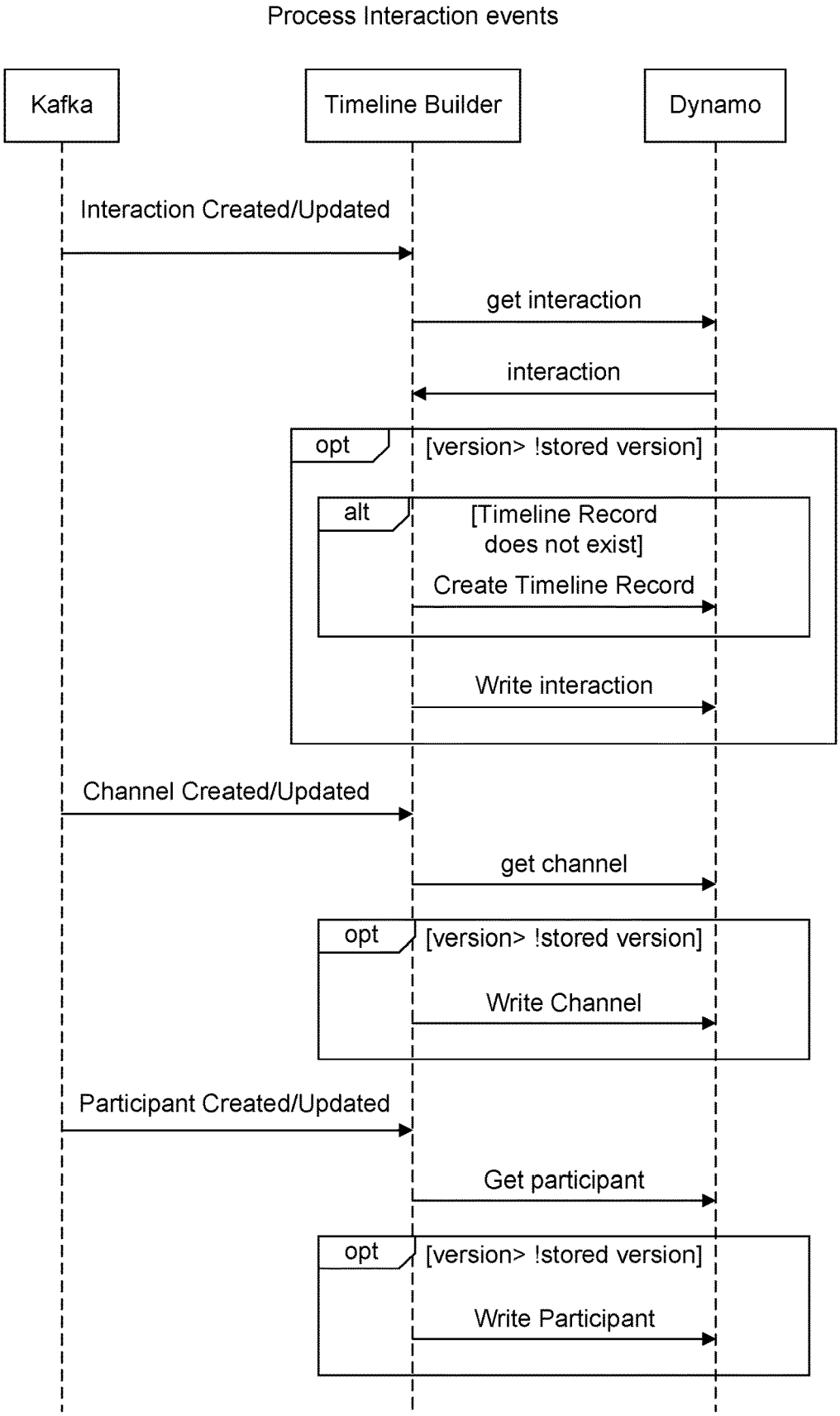

Process Interaction events
Kafka
Timeline Builder
Dynamo
Interaction Created/Updated
get interaction
interaction
opt
[version> !stored version]
alt
[Timeline Record does not exist]
Create Timeline Record
Write interaction
Channel Created/Updated
get channel
opt
[version> !stored version]
Write Channel
Participant Created/Updated
Get participant
opt
[version> !stored version]
Write Participant

FIG. 8

Name	Type/Values	Optional	Description	Has PII
sid	sid	N	The system generated unique string used to identify the TimelineRecord	N
version	integer	N	Entity version used to deduplicate records	N
instance_sid	sid	N	Flex service instance sid	N
end_customer_sid	string	N	Aion customer profile sid/ Only customers with customer_sid will be shown in the timeline	N
date_created	date_time<ISO 8601>	N	Denotes when the Timeline Record was created on the DB	N
date_updated	date_time<ISO 8601>	N	Denotes when the Timeline Record was updated on the DB	N

FIG. 9

Name	Type/Values	Optional	Description	Has PII
sid	sid	N	Interaction SID	N
version	integer	N	Entity version used to deduplicate events	N
data_type	String Enum	N	The type of data on the timeline record Interaction External	N
status	string enum	N	Interaction status: ACTIVE, INACTIVE, CLOSED	N
topics	Array of sids	N	List of topic sids of the interaction	N
date_created	date_time<ISO 8601>	N	Denotes when the entity was created	N
date_updated	date_time<ISO 8601>	N	Denotes when the entity was most recently updated	N

FIG. 10

Name	Type/Values	Optional	Description	Has PII
sid	sid	N	sid	N
version	integer	N	Entity version used to deduplicate events	N
instance_sid	sid	N	Flex service instance sid	N
interaction_sid	sid	N	Interaction sid	N
type	String enum	N	Channel type: (EMAIL, VOICE, SMS, WHATSAPP,WEB UNKNOWN)	N
initiated_by	String enum	N	InitiatedBy: The actor that created this InteractionChannel. (CUSTOMER, USER, API, UNKNOWN)	
status	String enum	N	Participant status: (SETUP, ACTIVE, INACTIVE, CLOSING, CLOSED)	N
date_created	date_time<ISO 8601>	N	Denotes when the entity was created	N
date_updated	date_time<ISO 8601>	N	Denotes when the entity was most recently updated	N

FIG. 11

Name	Type/Values	Optional	Description	Has PII
sid	sid	N	sid	N
version	integer	N	Entity version used to deduplicate events	N
instance_sid	sid	N	Flex service instance sid	N
interaction_sid	sid	N	Interaction sid	N
interaction_channel_sid	sid	N	Channel sid	N
type	String enum	N	Participant type: (CUSTOMER, EXTERNAL, UNKNOWN, USER)	N
status	String enum	N	Participant status: (JOINING, JOINED, LEFT, FAILED)	N
name	string	N	Participant name, PII	Y
date_created	date_time<ISO 8601>	N	Denotes when the entity was created	N
date_updated	date_time<ISO 8601>	N	Denotes when the entity was most recently updated	N

FIG. 12

Name	Type/Values	Optional	Description	Has PII
sid	sid	N	The system generated unique string used to identify the ExternalRecord	N
version	integer	N	Entity version used to deduplicate record	N
friendly_name	friendly_name_descripton	Y	Name of the record, ex: "Lease 1234 signed"	Y
description	sid	Y	ExternalRecord description	Y
date_created	date_time<ISO 8601>	N	Denotes when the entity was created	N
date_updated	date_time<ISO 8601>	N	Denotes when the entity was most recently	N
			updated	

FIG. 13

Request Headers

I-Twilio-Auth-Account	sid	The Twilio Account sid
I-Twilio-Request-Id	sid	The Twilio REST API Request sid

Body:

date_created_record	sid	Date when the record was created
topic_type_sids	Array of sids	The list of topics that the record is associated with
type	Enum	TimelineRecord type, only external record will be supported "external"
friendly_name	string	Short description of the record: "Contract signed" "Terms and conditions accepted"
description	string	A longer description for the record which might help the representative to get context without opening other sources: "Contract XXXX signed regarding to the house XXXX"

FIG. 14

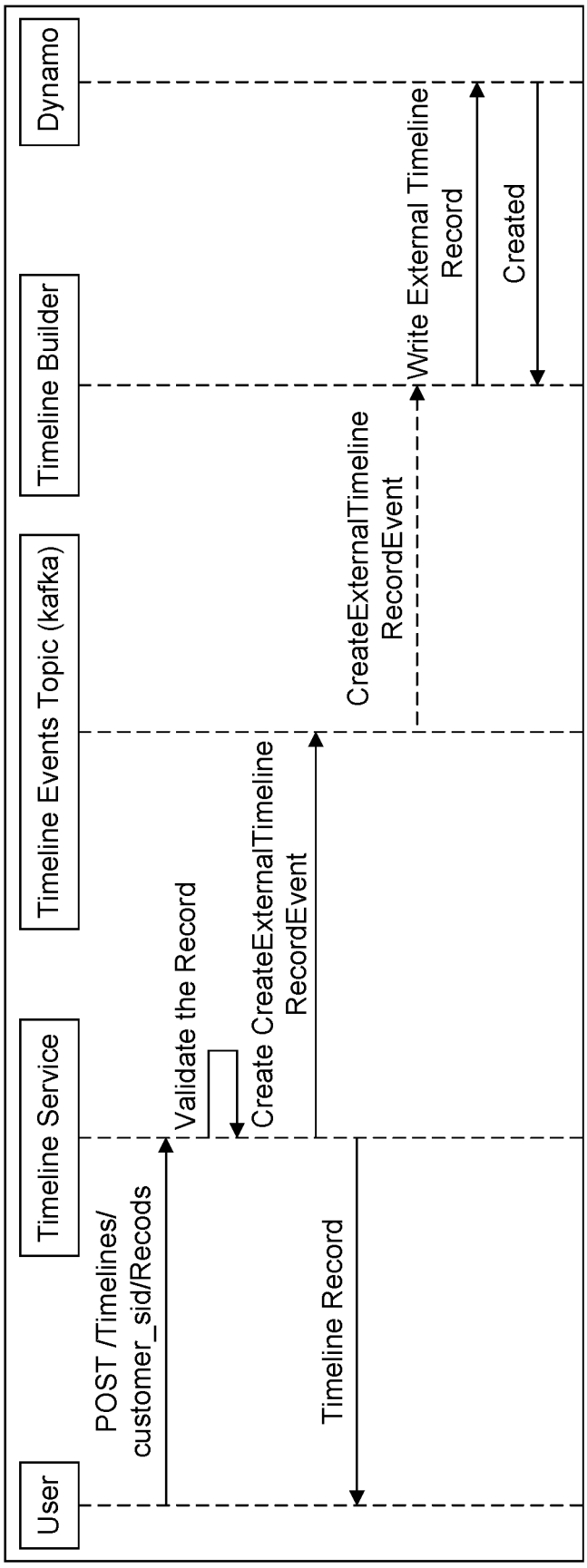

User
Timeline Service
Timeline Events Topic (kafka)
Timeline Builder
Dynamo
POST /Timelines/ customer_sid/Records
Validate the Record
Create CreateExternalTimeline RecordEvent
CreateExternalTimeline RecordEvent
Write External Timeline Record
Created
Timeline Record

FIG. 15

```
@startuml
!theme plain from
https://raw.githubusercontent.com/plantuml/plantuml/master/themes


package "Timeline" {

    class TimelineEvent {
      *sid: SID
      *version: INTEGER
      *instance_sid: SID
```

FIG. 16A

```
    *end_customer_sid: SID
    *date_created _event: DATETIME
    topic_type_sids: ARRAY
    *event_type: ENUM (INTERACTION, EXTERNAL)
    *event_data_sid: SID
    *date_created: DATETIME
    date_updated: DATETIME
}

class Topic {
    *sid: SID
    *version: INTEGER
    *date_created: DATETIME
    date_updated: DATETIME
}

class InteractionData {
    *sid: SID
    *version: INTEGER
    *instance_sid : SID
    *status : ENUM (ACTIVE, INACTIVE, CLOSED)
    *date_created : DATETIME
    date_updated: DATETIME
}

class Channel {
    *sid: SID
    *version: INTEGER
    *instance_sid: SID
    *interaction_sid: SID
    *type: ENUM (EMAIL, VOICE, SMS, WHATSAPP, WEB, UNKNOWN)
    *status: ENUM (SETUP, ACTIVE, INACTIVE, CLOSING, CLOSED)
    date_created: DATETIME
    date_updated: DATETIME
}

class Participant {
    *sid: SID
    *version: INTEGER
    *instance_sid: SID
    *interaction_sid: SID
    *interaction_channel_sid: SID
    *type: ENUM (CUSTOMER, EXTERNAL, UNKNOWN, USER)
```

FIG. 16B

```
        *status: ENUM (JOINING, JOINED, LEFT, FAILED)
        *user_sid: SID
        *date_created: DATETIME
        *date_updated: DATETIME
    }

    class ExternalData {
        *sid: SID
        *version: INTEGER
        *instance_sid: SID
        *customer_sid: SID
        description: STRING
        *date_created: DATETIME
        date_updated: DATETIME
    }

TimelineEvent ||--o| InteractionData
TimelineEvent ||--o| ExternalData
InteractionData ||--|{ Channel
InteractionData ||--o{ Topic

Channel ||--|{ Participant
@enduml
```

FIG. 16C

```
title get Timeline Records

participant User
participant Timeline Service
participant Memory store
participant Dynamo
participant User Service

deactivate Timeline Service
entryspacing 0.7
User- > Timeline Service: GET  /Timelines/{end_customer_sid}/Records
activate Timeline Service
Timeline Service-> Dynamo:Get Timeline Records:\n(Interactions, Channels,
Participants)

activate Dynamo
Timeline Service< -Dynamo:Interactions, Channels, Paticipants
deactivate Dynamo
Timeline Service- >Timeline Service:Build the Timeline Record with
Interactions
loop for each participant user sid
Timeline Service- >User Service:get User by ID
activate User Service
Timeline Service< -User Service:User
deactivate User Service

   Timeline Service - >Memory store:store the name
   Timeline Service - >Timeline Service: add the name to participants list in
   the Timeline Records list
   end

   Timeline Service - >User: List of Timeline Records
   deactivate Timeline Service
```

FIG. 17A

```
title get Timeline Records

participant User
participant Timeline Service
participant Memory store
participant Dynamo
participant User Service

group Initialize
activate Timeline Service
   Timeline Service- >Timeline Service: Initialize
Timeline Service- >Dynamo : Get all users sids
activate Dynamo
Timeline Service< -Dynamo:List of user SIDs
deactivate Dynamo
   loop for each user sid
   Timeline Service- > User Service:  get the name from the ID
activate User Service
---------------------------------------------------------------
Timeline Service< -User Service:Name of user
deactivate User Service
   Timeline Service- >Memory store: store the name in memory
deactivate Timeline Service
   end
end

deactivate Timeline Service
User- >Timeline Service: GET /timeline/{end_customer_sid}/Records
activate Timeline Service
Timeline Service- >Dynamo: Get Timeline Records:\n(Interactions,
Channels, Participants)

activate Dynamo
Timeline Service< -Dynamo:Interactions, Channels, Participants
deactivate Dynamo
Timeline Service- >Timeline Service:Build the Timeline Record with
Interactions
loop for each participant user sid
   Timeline Service- >Memory store: get the name from the ID
   Timeline Service- >Timeline Service: add the name to participants list in
the  timeline Record list
end
Timeline Service- >User: List of timeline records
deactivate Timeline Service
```

FIG. 17B

```
title process Interaction events

participant Kafka
participant Timeline Builder
participant Dynamo

entryspacing 0.9
Kafka- >>Timeline Builder:Interaction Created/Updated\n
Timeline Builder- >Dynamo:get interaction
Timeline Builder< -Dynamo:interaction
opt version  >  !stored version
alt Timeline Record does not exist
Timeline Builder- >Dynamo:Create Timeline Record
end
   Timeline Builder- >Dynamo:Write interaction
end

Kafka- >>Timeline Builder:Channel Created/Updated
Timeline Builder- >Dynamo:get channel
opt version  >  !stored version
   Timeline Builder- >Dynamo:Write Channel
end
Kafka- >>Timeline Builder:Participant Created/Updated
Timeline Builder- >Dynamo:Get participant
opt version  >  !stored version
   Timeline Builder- >Dynamo:Write Participant
end
```

FIG. 17C

```
title External Timeline Record Creation

participant User
participant Timeline Service
participant Timeline Events Topic  (Kafka)
participant Timeline Builder
participant Dynamo

User- >Timeline Service: POST  /Timelines/customer_sid/Recods
Timeline Service- >Timeline Service:Validate the Record
Timeline Service- >Timeline Events Topic  (Kafka) :Create
CreateExternalTimelineRecordEvent
User< -Timeline Service:Timeline Record
Timeline Events Topic (Kafka)-- >Timeline
Builder:CreateExternalTimelineRecordEvent
Timeline Builder- >Dynamo:Write External Timeline Record
Timeline Builder< -Dynamo:Created
```

FIG. 18

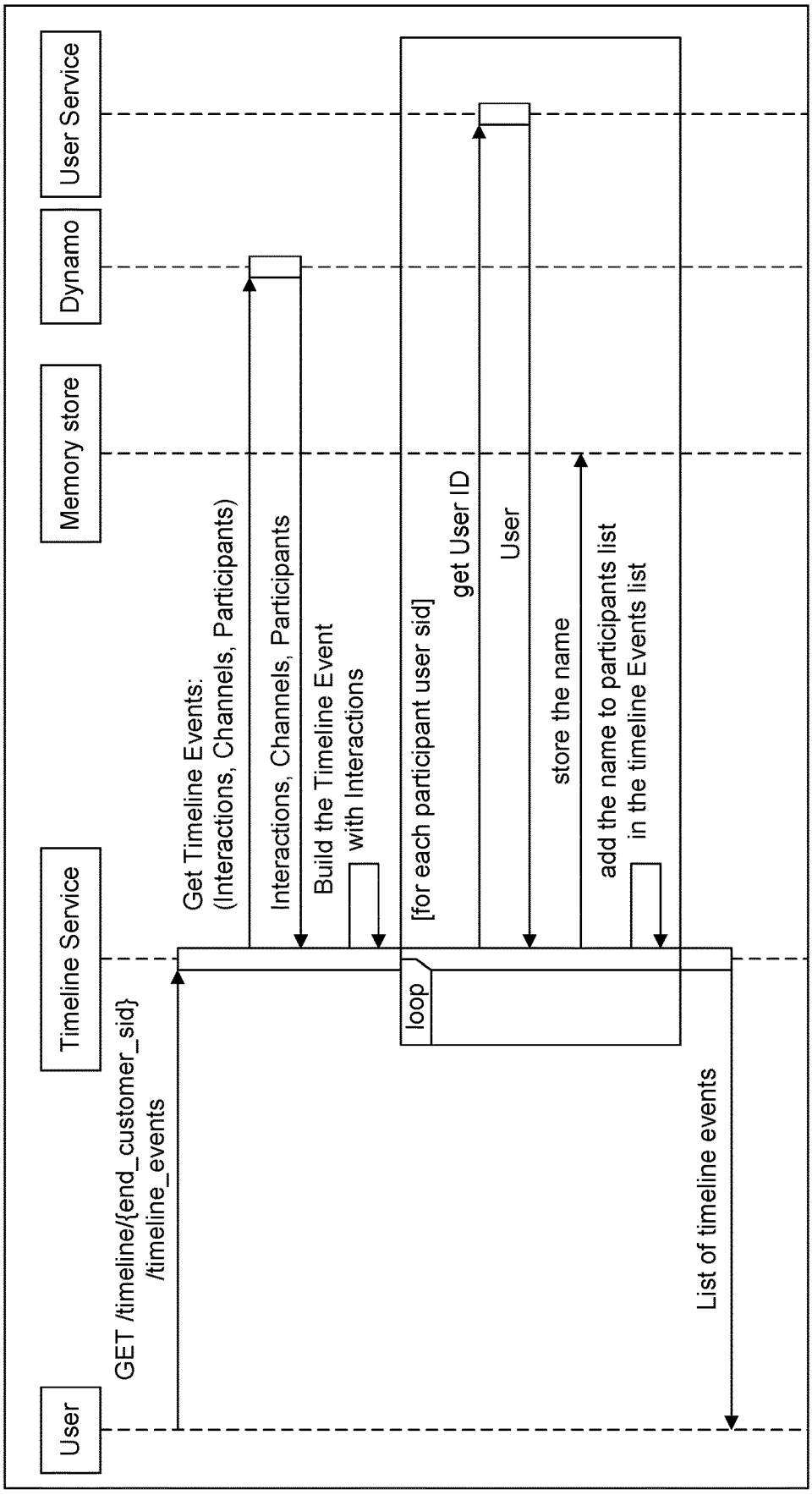

User
Timeline Service
Memory store
Dynamo
User Service
GET /timeline/{end_customer_sid}
/timeline_events
Get Timeline Events:
(Interactions, Channels, Participants)
Interactions, Channels, Participants
Build the Timeline Event
with Interactions
loop
[for each participant user sid]
get User ID
User
store the name
add the name to participants list
in the timeline Events list
List of timeline events

FIG. 19

```
SELECT from TimelineRecord
WHERE instance_sid = ?
AND end_customer_sid = ?
```

```
{ SID: EC001
    [
        { date: 1/1/23, data_type: INTERACTION },
        { date: 2/1/23, data_type: EXTERNAL }
    ]
}
```

FIG. 20

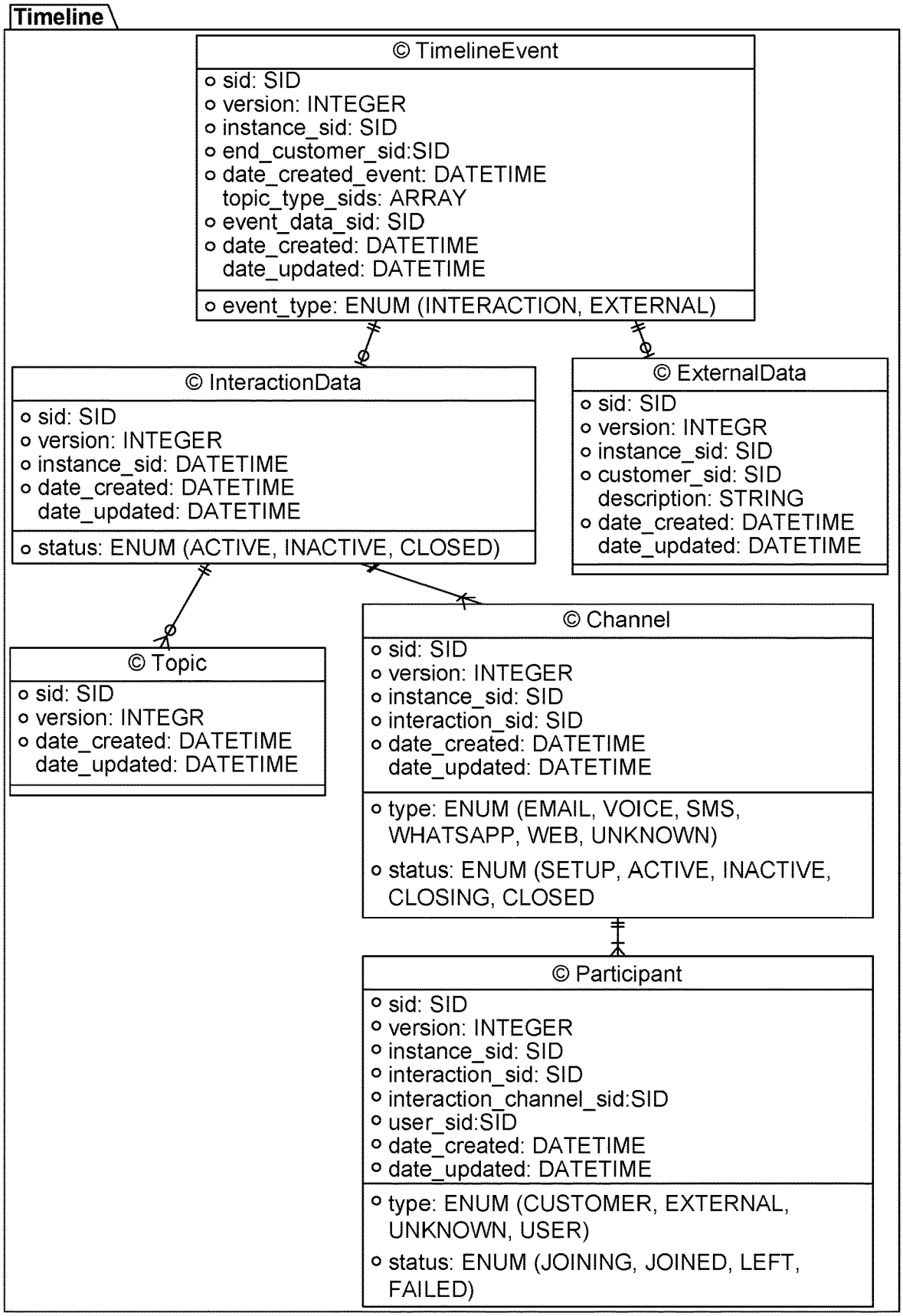

Timeline
© TimelineEvent
sid: SID
version: INTEGER
instance_sid: SID
end_customer_sid:SID
date_created_event: DATETIME
topic_type_sids: ARRAY
event_data_sid: SID
date_created: DATETIME
date_updated: DATETIME
event_type: ENUM (INTERACTION, EXTERNAL)
© InteractionData
sid: SID
version: INTEGER
instance_sid: DATETIME
date_created: DATETIME
date_updated: DATETIME
status: ENUM (ACTIVE, INACTIVE, CLOSED)
© ExternalData
sid: SID
version: INTEGR
instance_sid: SID
customer_sid: SID
description: STRING
date_created: DATETIME
date_updated: DATETIME
© Topic
sid: SID
version: INTEGR
date_created: DATETIME
date_updated: DATETIME
© Channel
sid: SID
version: INTEGER
instance_sid: SID
interaction_sid: SID
date_created: DATETIME
date_updated: DATETIME
type: ENUM (EMAIL, VOICE, SMS, WHATSAPP, WEB, UNKNOWN)
status: ENUM (SETUP, ACTIVE, INACTIVE, CLOSING, CLOSED
© Participant
sid: SID
version: INTEGER
instance_sid: SID
interaction_sid: SID
interaction_channel_sid:SID
user_sid:SID
date_created: DATETIME
date_updated: DATETIME
type: ENUM (CUSTOMER, EXTERNAL, UNKNOWN, USER)
status: ENUM (JOINING, JOINED, LEFT, FAILED)

FIG. 21

Index	*Field*	Example Value	Type	Description
PK hash	*account_ sid#parent_sid*	*AC00000000000000000000000 0000000001*#KD00000000000 0000000000000000001	Sting	Partition key-By account and interaction_sid. Note all entity types such as channels and participants have a parent interaction_sid reference. That parent reference is the interaction_sid being stored here.
PK range	*entity_sid*	KE00000000000000000000000 00000001	String	In this example it's a channel_sid. All sub entities of an Interaction have the Interaction_sid which is part of our primary key.
GSI *summary _index*	*account_sid# profile_connec t_sid#interact on_sid*	*AC00000000000000000000000 0000000001*#YI000000000000 0000000000000000001#KD000 0000000000000000000000000 001	String	Sparse index used to retrieve Interaction Summaries. Note we're just retrieving type Interaction items and not the channels and other entities whose parent an interaction.
	status_id#date _modified	300#2021-02-07T 10:37:00Z	String	Regular attribute used as a sort key of GSI. All Interactions for a profile ordered by status then by date modified
GSI	*instance*	GO00000000000000000000000 000000001	String	Indexed identifier for logically separating data. For Aion Interactions this would be the *flex_instance_sid.* Indexing this field allows for retrieving and exporting all profiles. For non Aion instances, can this be defined via the Marketplace config for the CDP attached?
	timestamp	1677819600	Integer	Timestamp for TTL retention peroid.
	parent_sid	KD00000000000000000000000 00000001		*Interaction_sid* from Aion entities. We use a generic name for the field.

Ⓐ

FIG. 22

Ⓐ

	profile_connect_sid	YI00000000000000000000000000000001	String	Can be missing for unknown events and added later.
	status_id	300 Integer		Only added to Interaction types. ACTIVE= 300, INACTIVE=200, CLOSED=100
	date_modified	2021-02-07T10:37:00Z	String	
	version	1 Integer		For Aion, change events all come with a version number. We assume that the data property of each item entity is mutable and that version is incremented. Versioned data establishes an order of occurrence, disregarding out of order changes.
	attr	{...}	JSON String	Is the JSON serialized as binary or string?
.. .	.. .	.. .	.. .	.. .

FIG. 22(Continued)

partition_key	sort_key	gsi1	gsi1_sort_key	instance_sid	timeline_record_sid	parent_sid	status_id	profile_connect_sid	data_modified	version	attr	ttl
AC00001#KD00001	KD00001	AC00001#YI0000	300#2021-02-07T1	GO00001	ZZ0001	KD00001	300	YI00001	2021-02-07T10:37	1	{date_created: 2021-02-07T10:37:00Z	1677819600

FIG. 23

	1	0:37:00Z						:00Z		, status: ACTIVE, data_ type: INTERACTION, topic_sids: [topic1_sid, topic2_sid] }

Channel

partition_ key	sort_ key	instance_ sid	timeline_ record_sid	parent_ sid	profile_ connect_sid	date_ modified	version	attr	ttl
AC00001 # KD00001	KE00001	GO0000001	ZZ0001	KD000001	YI00001	1/12/2023 12:13:22	1	{status: ACTIVE, type: VOICE, data_created: 1/12/2023 12:13:22, initiated_by: CUSTOMER }	1677819600

Participant

partition_ key	sort_ key	instance_ sid	timeline_ record_sid	parent_ sid	profile_ connect_sid	date_ modified	version	attr	ttl
AC00001 # KD00001	KF00001	GO0000001	ZZ0001	KD000001	YI00001	1/12/2023 12:13:22	1	{user_sid:US0001, type: CUSTOMER, status: JOINED, date_created: 1/12/2023 12:13:22, channel_sid: KE0001 }	1677819600

FIG. 24

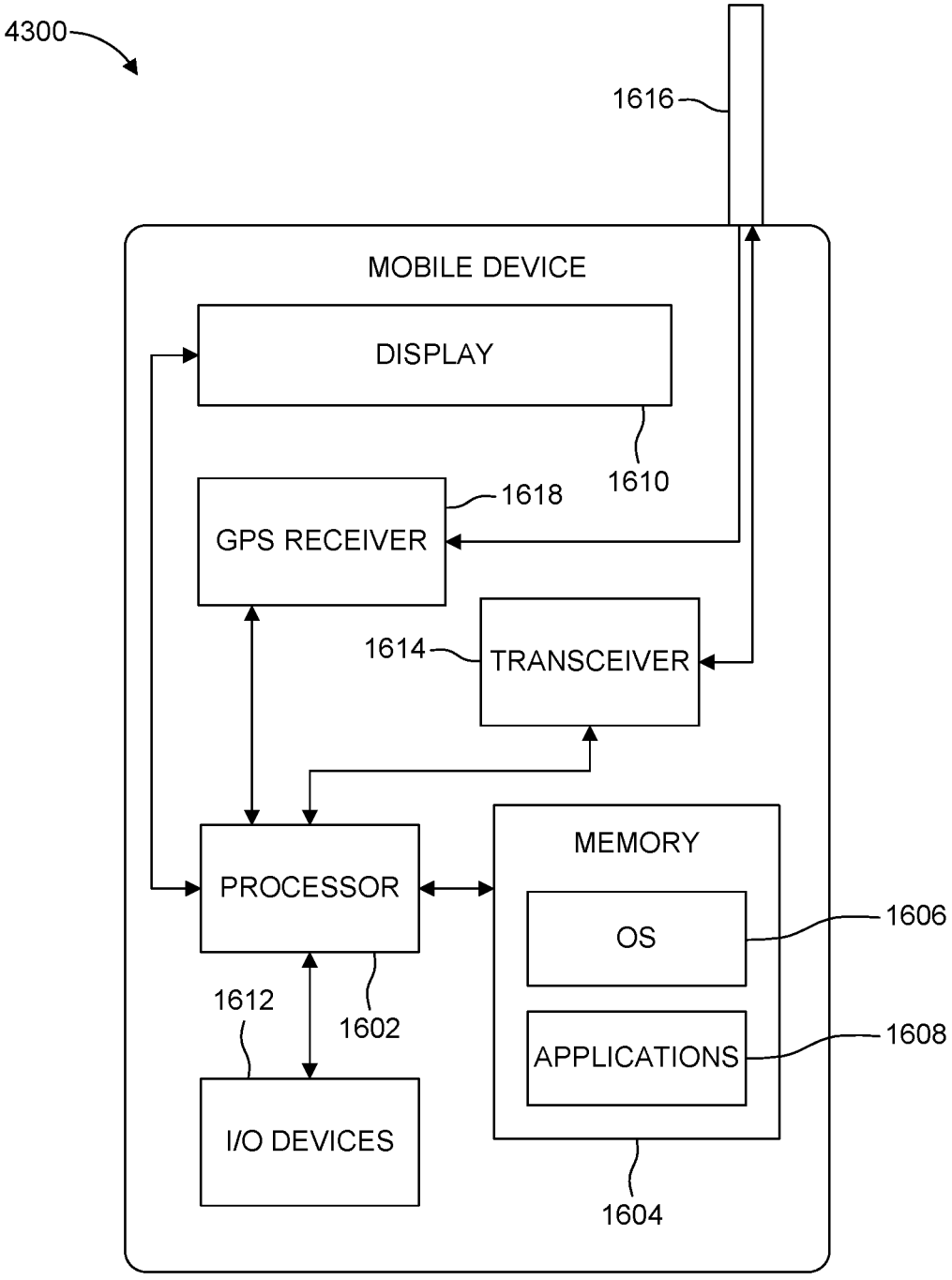

4300
1616
MOBILE DEVICE
DISPLAY
1610
1618
GPS RECEIVER
1614
TRANSCEIVER
PROCESSOR
MEMORY
OS
1606
APPLICATIONS
1608
1612
1602
I/O DEVICES
1604

FIG. 25

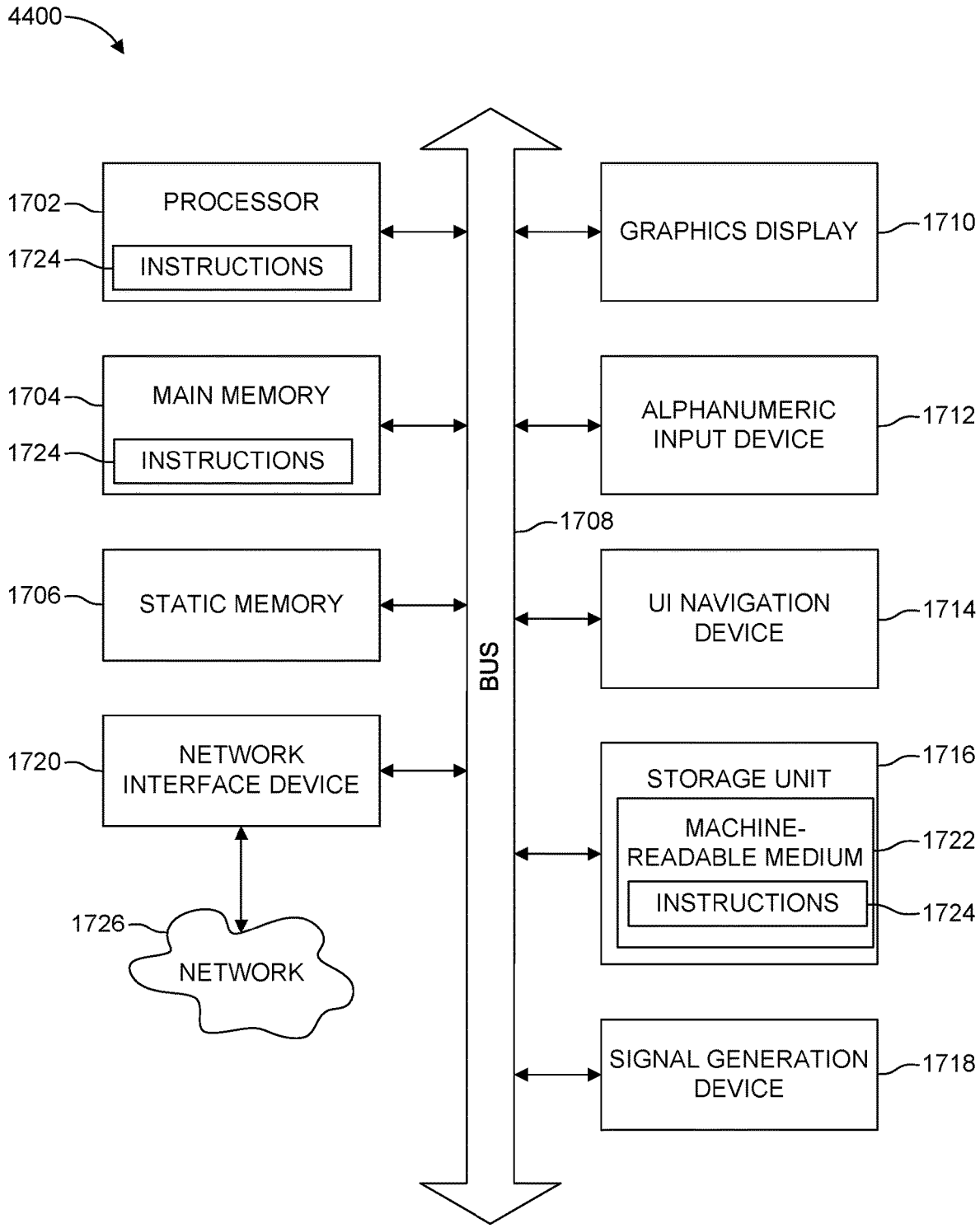

4400
1702
PROCESSOR
1724
INSTRUCTIONS
1704
MAIN MEMORY
1724
INSTRUCTIONS
1706
STATIC MEMORY
1720
NETWORK INTERFACE DEVICE
1726
NETWORK
BUS
1708
GRAPHICS DISPLAY
1710
ALPHANUMERIC INPUT DEVICE
1712
UI NAVIGATION DEVICE
1714
STORAGE UNIT
1716
MACHINE-READABLE MEDIUM
1722
INSTRUCTIONS
1724
SIGNAL GENERATION DEVICE
1718

FIG. 26

PROGRAMMABLE TIMELINE FEATURE FOR COMMUNICATION SYSTEMS

CLAIM OF PRIORITY

[0001] This application claims the benefit of U.S. Provisional Application No. 63/363,935, filed Apr. 29, 2022, which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The disclosed subject matter relates generally to the technical field of cloud-based software-as-service (SaaS) architectures and, in one specific example, to a programmable timeline feature for communication systems.

BACKGROUND

[0003] Entities, such as public or private corporations, have unique needs for managing their customer engagement workflows. Current solutions require businesses to make significant investments in hardware and offer only prepackaged implementations that are limiting.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Some embodiments are illustrated by way of example and not limitation in the figures of the accompanying drawings.

[0005] FIG. **1** is a network diagram depicting a cloud-based SaaS system within which various example embodiments may be deployed.

[0006] FIG. **2** is a block diagram illustrating example modules of the service(s) of FIG. **1**.

[0007] FIG. **3** is a block diagram depicting an example architecture and data flow for the system **100**.

[0008] FIG. **4** is a flow chart showing operations for using the Timeline API to get a Timeline GUI comprising a set of timeline entries.

[0009] FIG. **5** is a block diagram depicting a data flow through an example architecture of the system.

[0010] FIG. **6** is a flow chart depicting an example flow for listing all of the timeline records ordered by created date.

[0011] FIGS. **7**A-**7**C are listings of an example response to a request.

[0012] FIG. **8** is a flow chart depicting an example of processing interaction events with the Timeline Builder.

[0013] FIG. **9** is a table describing an example schema for a timeline record.

[0014] FIG. **10** is a table describing an example schema for interaction data.

[0015] FIG. **11** is a table describing an example schema for a channel.

[0016] FIG. **12** is a table describing an example schema for a participant.

[0017] FIG. **13** is a table describing an example schema for external records. In example embodiments, external records include timeline records generated by an entity using the Timeline service.

[0018] FIG. **14** is a table of example request headers and an example message body for adding an external record to a timeline.

[0019] FIG. **15** is a sequence diagram showing an example of how an external timeline record may be created.

[0020] FIGS. **16**A-**16**C is a listing of an example entity relationship diagram.

[0021] FIG. **17**A is a listing of getting timeline records with cache.

[0022] FIG. **17**B is a listing of getting timeline records with initialization and events.

[0023] FIG. **17**C is a listing of processing interaction events.

[0024] FIG. **18** is a listing of creating an external timeline event.

[0025] FIG. **19** is a flowchart showing example get timeline events.

[0026] FIG. **20** is a listing of an example select statement.

[0027] FIG. **21** is a block diagram showing a normalized view of an example logical model of the data a timeline may have, including the entity relations.

[0028] FIG. **22** is a table showing an example index.

[0029] FIGS. **23**-**24** are tables showing examples of how data (e.g., interaction, channel, and participant data) will go into a database.

[0030] FIG. **25** is a block diagram of an example mobile device, according to an example embodiment.

[0031] FIG. **26** is a block diagram of a machine in the example form of a computer system within which instructions for causing the machine to perform any one or more of the methodologies discussed herein may be executed.

DETAILED DESCRIPTION

[0032] In the following description, for purposes of explanation, numerous specific details are set forth in order to provide an understanding of various embodiments of the present subject matter. It will be evident, however, to those skilled in the art that various embodiments may be practiced without these specific details.

[0033] A system, including a software development kit (SDK) (e.g., also referred to herein as "the system" or "the service(s)"), is disclosed that allows one or more timelines, as described herein, to be generated, stored, accessed, modified, edited, customized, and presented as a visual representation in one or more graphical user interfaces (GUIs), as described herein. Such timelines may be integrated into one or more customized workflows corresponding to one or more end customers of an entity, such as an end customer identified (e.g., via a customer identity service) as being associated with an incoming communication with the entity. Such engagement workflows may include one or more workflows through a contact center, such as for an end customer journey associated with one or more topics corresponding to an interaction, as described herein. In example embodiments, a topic represents a tag or category selected from a classification system for classifying an interaction, and may represent an intent of an end customer. In example embodiments, the timeline and tools for generating, storing, accessing, modifying, customizing, and presenting the timeline may be made accessible (e.g., programmatically, via an API, and/or via one or more GUIs) to one or more end customers of the entity, administrators of the entity, representatives of the entity, or systems, including machine-learning systems, chatbots or other systems configured to automate tasks (e.g., including workflow tasks), and so on, as described herein.

[0034] In example embodiments, a timeline includes information pertaining to each of an end customer's historical interactions with an entity, such as a business entity.

[0035] The SDK supports generating specialized and/or customized timeline data objects; adding, editing, and

removing data, including data pertaining to communication data and communication metadata (e.g., corresponding to voice or digital communications managed by a communication platform) and/or end customer data (e.g., information about an entity's end customers managed by one or more systems of the entity, such as Customer Data Platform (CDP) systems of the entity) using the timeline data objects; and/or surfacing at least one or more subsets or summaries of the one or more subsets of the data objects in one or more custom applications, such as one or more contact center applications or other communication applications.

[0036] In example embodiments, one or more custom applications may call one or more APIs of a timeline feature (e.g., implemented as a software as an application (SaaS) or other service) to access a timeline data object (e.g., corresponding to the timeline of one or more interactions of an end customer) to retrieve data pertaining to an interaction of an end customer. In example embodiments, the timeline data object is specially stored and/or configured to provide efficient real-time access all or one or more subsets of the data stored in or referenced from the timeline object for surfacing in real-time in a custom graphical user interface (GUI), such as a custom user interface configured to present a summary of the data to an agent or representative in a contact center. The representative may then view the summary of the data when determining whether to accept an invitation to process the interaction. If the representative chooses not to accept the invitation, the interaction may be routed to one or more additional representatives, who, in turn, may view a visual representation of the timeline data object, including the summary of the data, and make a decision of whether to accept the invitation.

[0037] Upon acceptance of an invitation by a representative, further information pertaining to the timeline may be populated into the user interface of the representative to allow the representative to process the interaction efficiently. In example embodiments, the information that is presented in the representative's user interface at each stage of a customer's journey (e.g., a first stage occurring before the representative accepts the invitation, a second stage occurring while the representative is handling or processing the interaction, and a third stage occurring when the representative is done handling or processing the interaction) is customizeable and/or configurable by systems of the representative, the entity, or the customer, or any combination of such systems (e.g., using the timeline API or options available within one or more user interfaces presented to the users via the systems of the users). Thus, for example, a summary of various data fields may be presented in a visual representation of a timeline data object in a user interface during the first stage and more detailed information may be presented in the user interface upon accepting of an invitation by a representative in the second stage. In example embodiments, the more detailed information may be based on various factors, such as an association of one or more topics with communication data, end customer data, and/or metadata pertaining to the interaction that is being handled or processed.

[0038] In example embodiments, data pertaining to previous communications of the end customer, such as data maintained and stored in data stores managed by a communication platform, is merged or aggregated with data pertaining to the end customer, such as the end customer's profile data or custom data about end customers, that is managed stored in systems managed by the entity or accessible from systems managed by the entity separately from the communication platform.

[0039] In example embodiments, the representative may specify one or more values of one or more new data items and modify the timeline data object corresponding to an interaction such that the values are merged or added to the timeline data object. The one or more new data items may then be accessible to subsequent representatives who access the timeline data object, such as when an interaction is routed to the one or more subsequent representatives for handling or processing the interaction. In example embodiments, one or more of the data values stored with respect to the timeline data object may be references to data items rather than actual values of data items. Thus, for example, end customer data, such as end customer data that is stored in systems managed by the entity separately from the communication platform. may be associated with the timeline data object without requiring the actual values of the data items to be stored in the timeline data object. In example embodiments, the use of references may reduce the regulatory burden on the data owner.

[0040] In example embodiments, an interaction represents a customer contact. Interactions may comprise a single work record for some or all of the ongoing and past conversations over one or multiple channels between entity representative (s), system(s), and contact(s), and/or data needed to provide personalized and contextual customer engagement. Interactions enable developers to build structured, directed communication workflows with the capability to address reporting and audit needs. In example embodiments, in a contact center context, an interaction may represent a contact center work item that needs to be processed by the contact center representative

[0041] An interaction may represent a customer engagement. In example embodiments, it maps to an end-customer. An end-customer can participate in one or more simultaneous interactions.

[0042] An interaction may be omnichannel. For example, Interactions support synchronous channels (e.g., voice, video) and asynchronous channels (e.g., chat, messaging, email) supported by a communication platform.

[0043] An interaction supports custom channels. In example embodiments, this allows entities to route any piece of work in an organization while providing context and a holistic view of each touchpoint for a customer.

[0044] An interaction may be channel agnostic and/or it may be performed over 0 or more communication channels, including custom channels, and/or over a non-contiguous timeline.

[0045] Channels of an interaction may be agnostic to external, internal humans or systems as participants. At a given time, Interactions channels may have zero or more participants.

[0046] An interaction may get offered (e.g., via an invitation) to one or many users or systems, such as contact center representatives or machine-learning systems, and the users or systems may accept or decline the interaction for processing.

[0047] An interaction may have a state which includes active, inactive (similar state parked or snoozed), and closed. In example embodiments, once closed, an interaction cannot reactivate. It can be closed by humans or systems.

[0048] An interaction may have a well-defined start and end signal and duration with a disposition.

[0049] An interaction may include a container to log and retrieve real-time and historical conversations and events. Logged interactions may be stored in a data object specially configured to represent the interaction data over a timeline.

[0050] An interaction may serve as a source of truth for data required to derive metrics, calculate KPIs, and/or route the interaction to the right available entity representative.

[0051] An interaction may get grouped in zero or more views. Views group interaction by attributes such as time, system state, and attached tags. Views are versioned and have a known refresh rate to support real-time and historical use cases. The views may be surfaced in one or more user interfaces.

[0052] An interaction may expose an interface for orchestrating customer engagement workflows such as routing, channel control operations such as hold, transfer, barge, whisper, and/or post-wrap-up actions, etc.

[0053] An interaction may include a resource to handle customer contact and/or a container for omnichannel conversations. In example embodiments, an interactions exposes an interface to manage channels and participants attached to a customer contact.

[0054] An interaction may have one or more states. These states may include any of the following states: Closed: The interaction is considered to be complete and no further activity will happen. In example embodiments, a closed interaction cannot be reopened but additional context can be added to it. Inactive: The interaction becomes inactive if Activity pauses on it for some duration. A user or a system can put interaction in Inactive state. More activity can take place which will bring the interaction back to an active state. Active: The interaction is created in Active state.

[0055] In example embodiments, interactions are represented as a single work record for all the ongoing conversations over one or multiple channels between a business representative(s) or a system (such as IVR, chatbot, etc.) and the end customer. Data needed from different systems/applications to support one or more topics associated with each interaction and/or from other business backend operations that can bring information to help provide personalized, contextual customer engagement. Interactions may allow entities to build structured, directed communication workflows with capability to address reporting and audit needs. It supports voice and digital channels, including email as asynchronous channels, with messaging and email channels through unified omni-channel conversations. Interactions also support custom channels allowing entities to route any piece of work in an organization while providing context and a holistic view of each and every touch point for a customer.

[0056] The system provides a single source of truth about anything that has happened during or after the life cycle of an interaction. It includes a single data source for all real-time and historical data to derive metrics, calculate KPIs, and to route the interaction to the right available representative or system.

[0057] Behind Interactions sits a well-defined data model called an Interaction Detail Record (IDR). IDR is a representation of the Interactions from being created until closed. It is a cradle-to-grave view, highlighting who did what and when. Although IDR's may be defined to address specific contact center needs, they are not exclusively for Contact Centers. The IDR can address the requirements of any entity communicating in which internal users, such as representatives, communicate internally with each other or with outside users, such as customers.

[0058] Each interaction consists of participants that represent the parties in the interaction that contribute to the communication channel. Examples of participants are External (customer), User (e.g., agent), IVR, Queue, Bot, or System. Participants communicate using specific channels which are represented in the data model of Interactions. An interaction maintains a roster of one or more participants which each may be connected to one or more channels. With each state change, more granular parts demarcate noteworthy differences in each timespan, such as when the state of a voice channel goes from talking to on hold.

[0059] Parts are the atoms of an interaction. In example embodiments, the system may be configured to incrementally build a detailed record view in real/near-real-time as each part occurs. Each part contains a collection of attributes/dimensional data specific to that level of the detail record hierarchy (e.g., ANI/DNIS/email, state, queue id, team id, user id, direction, disposition code, disconnect reason, etc.). Each part may intrinsically carry specific context and information to support very granular drill-down.

[0060] For example, the agent parts convey relevant identifiers (e.g., queue, user, interaction, and disposition codes), channel actions (e.g. when did the agent put the customer on hold), and state changes (e.g. when an interaction becomes inactive). Another example is the system participants such as ACD/queue. For these participants, parts may contain a record of the routing decisions employed in order to get the interaction routed to an agent (e.g., skills-based routing). As a final example, the IVR/bot participant can also include important context (e.g. id/version of the bot employed in that interaction and which conditional/control branches the customer traversed and when, etc.).

[0061] In summary, every participant in an interaction may play a particular role and generate unique parts that are captured in an IDR for storage and further analysis.

[0062] In example embodiments, an interaction represents a single work item that needs to be processed. It represents a contact between an entity representative and a customer related to the same or a similar topic. Interactions are channel-agnostic and may represent voice, web chat, SMS, WhatsApp, Email, and so on, and can support multiple channels simultaneously. Interactions may support multiparty conversations.

[0063] In example embodiments, the disclosed API facilitates live message and file exchange between multiple participants on supported channels (SMS, MIMS, Whatsapp, in-app chat, email, and so on). The API may explicitly focus on facilitating active communication and may provide search tools that enable discovery of ongoing and archived conversations or communications, and include non-message "events" in the timeline like "participant added to the thread."

[0064] The customer journey is about the customer lifecycle and total context of the customer incorporating all the communications throughout the life time. A journey would be one or more interactions as well as other, non-interaction data. This view of a customer lifecycle spanning over several specific encounters (interactions) can give broader context to an agent about the customer and be used to build user interfaces.

[0065] Context Store is the storage of messages, conversations, events, timelines, and the relationships between these items on a timeline. Interactions may add to the Context Store, allowing customers to see journey information that includes interaction and non-interaction data.

[0066] In example embodiments, Interaction Detail Records are different from the context store in its near/real-time requirement, its structured data model, and its defined start and end-time.

[0067] In example embodiments, a touchpoint record or engagement record is a customer touchpoint is a record of any time a consumer interacts with a company, through an interaction, managed by an agent, an interaction managed by a bot, a web site, a CDP, or an application. From now on touchpoints and Timeline Records will be mentioned and they can be treated as the same term in the Timeline Service context.

[0068] In example embodiments, a timeline includes a list of touchpoints of an end customer in a company over a specific period of time. It typically includes key touchpoints. such as purchase or service interactions, support requests, and communication with the company. An end customer timeline is used to understand the end customer's experience and identify any pain points or areas for improvement.

[0069] In example embodiments, a timeline record includes a touchpoint that might represent any interaction of customers with the company using a communication platform or through their external systems.

[0070] In example embodiments, platform timeline records include a type of touchpoint or timeline record of any time a consumer interacts with an entity using internal communication platform services or external (e.g., CDP) services.

[0071] In example embodiments, interaction timeline records include a type of touchpoint or Timeline record of any time an end customer interacts with an entity using Interactions. An interaction might be managed by or surface to an agent, by a bot an IVR, etc., or any participant that can handle Interactions in the system.

[0072] In example embodiments, external timeline records include a type of touchpoint or Timeline record of any time an end customer interacts with an entity outside of a communications context; for example, the customer submitting a loan application, signing a contract with the company; or interacting with external systems like CDPs, company websites, or company systems, etc.

[0073] A major pain point for end customers is repeating themselves with each interaction or transfer. In order to remove this repetition, the intention is to provide representatives, systems, and other parties with a timeline that lists previous interactions. The timeline may be retrieved for the End-customer when they connect with a representative and may be drawn from interactions (including communication channels and participants), customer profiles, and topics.

[0074] In example embodiments, a custom application inputs an end-customer identifier (e.g., via the API) and receives a timeline as output, such as a timeline data object and/or a rendered timeline. In example embodiments, the timeline services are only available to authenticated actors with authorized roles (e.g., using roles and permissions as defined by the entity).

[0075] In example embodiments a method of processing one or more interactions using a timeline is disclosed. Data pertaining to a plurality of interactions is stored in a timeline record in real-time when each of the plurality of interactions occurs. Each of the plurality of interactions is associated with one or more topics and an identifier of an end customer. Each of the one or more topics identifying an intent of the end customer. A request for a subset of the data that corresponds to an additional interaction of the end customer is received via an API. The subset identified in one or more parameters of the API. In response to the request, the subset of the data is provided for surfacing in a graphical user interface such that one or more representatives can make a determination of whether to accept an invitation to process the additional interaction.

[0076] FIG. **1** is a network diagram depicting a system **100** within which various example embodiments may be deployed. A networked system **102**, in the example form of a cloud computing service, such as Microsoft Azure or other cloud service, provides server-side functionality, via a network **104** (e.g., the Internet or Wide Area Network (WAN)) to one or more endpoints (e.g., client machines **110**). FIG. **1** illustrates client application(s) **112** on the client machines **110**. Examples of client application(s) **112** may include a web browser application, such as the Internet Explorer browser developed by Microsoft Corporation of Redmond, Washington or other applications supported by an operating system of the device, such as applications supported by Windows, iOS or Android operating systems. Each of the client application(s) **112** may include a software application module (e.g., a plug-in, add-in, or macro) that adds a specific service or feature to the application.

[0077] An API server **114** and a web server **116** are coupled to, and provide programmatic and web interfaces respectively to, one or more software services, which may be hosted on a software-as-a-service (SaaS) layer or platform **104**. The SaaS platform may be part of a service-oriented architecture, being stacked upon a platform-as-a-service (PaaS) layer **106** which, may be, in turn, stacked upon a infrastructure-as-a-service (IaaS) layer **108** (e.g., in accordance with standards defined by the National Institute of Standards and Technology (NIST)).

[0078] While the applications (e.g., service(s)) **120** are shown in FIG. **1** to form part of the networked system **102**, in alternative embodiments, the applications **120** may form part of a service that is separate and distinct from the networked system **102**.

[0079] Further, while the system **100** shown in FIG. **1** employs a cloud-based architecture, various embodiments are, of course, not limited to such an architecture, and could equally well find application in a client-server, distributed, or peer-to-peer system, for example. The various server applications **120** could also be implemented as standalone software programs. Additionally, although FIG. **1** depicts machines **110** as being coupled to a single networked system **102**, it will be readily apparent to one skilled in the art that client machines **110**, as well as client applications **112**, may be coupled to multiple networked systems, such as payment applications associated with multiple payment processors or acquiring banks (e.g., PayPal, Visa, MasterCard, and American Express).

[0080] Web applications executing on the client machine (s) **110** may access the various applications **120** via the web interface supported by the web server **116**. Similarly, native applications executing on the client machine(s) **110** may access the various services and functions provided by the applications **120** via the programmatic interface provided by

the API server **114**. For example, the third-party applications may, utilizing information retrieved from the networked system **102**, support one or more features or functions on a website hosted by the third party. The third-party website may, for example, provide one or more promotional, marketplace or payment functions that are integrated into or supported by relevant applications of the networked system **102**.

[0081] The server applications **120** may be hosted on dedicated or shared server machines (not shown) that are communicatively coupled to enable communications between server machines. The server applications **120** themselves are communicatively coupled (e.g., via appropriate interfaces) to each other and to various data sources, so as to allow information to be passed between the server applications **120** and so as to allow the server applications **120** to share and access common data. The server applications **120** may furthermore access one or more databases **126** via the database servers **124**. In example embodiments, various data items are stored in the database(s) **126**, such as the system's data items **128**. In example embodiments, the system's data items may be any of the data items described herein.

[0082] Navigation of the networked system **102** may be facilitated by one or more navigation applications. For example, a search application (as an example of a navigation application) may enable keyword searches of data items included in the one or more database(s) **126** associated with the networked system **102**. A client application may allow users to access the system's data **128** (e.g., via one or more client applications). Various other navigation applications may be provided to supplement the search and browsing applications.

[0083] FIG. **2** is a block diagram illustrating example modules of the system's service(s) **120**. In example embodiment, an SDK module **202** is configured to provide access to one or more APIs for performing one or more tasks, as described herein. An interactions module **204** is configured to provide access to one or more APIs for managing interactions or interactions services and/or performing one or more related tasks, as described herein. A channels module **206** is configured to provide access to one or more APIs for managing channels or channel services and/or performing one or more related tasks, as described herein. A users (or participant) module **208** is configured to provide access to one or more APIs for managing users or user services and/or performing one or more related tasks, as described herein. A timeline module **210** is configured to provide access to one or more APIs for managing timelines or timeline services and/or performing one or more related tasks, as described herein. A dynamic graphical user interface module **212** is configured to cause one or more user interfaces to be presented on one or more devices, as described herein.

[0084] FIG. **3** is a block diagram depicting an example architecture and data flow for the system **100**.

[0085] The intended design leverages a customer data platform (CDP) system. The CDP platform can host end-customer profiles and has identity resolution capabilities that can disambiguate multiple customer entities. The CDP may provide end customer profile data and some historical data on end-customers. Other sources of data include timeline events and interactions made on communication channels of communication platform that is separate from the CDP, as described herein.

[0086] In example embodiments, entities gather their end customer profiles into the CDP. In example embodiments, when an interaction with an end customer commences the profile, topics and interaction details are combined before the Timeline service is initialized. To reduce latency the timeline assembles as much as possible while the customer waits (e.g., in a lobby).

[0087] This approach de-couples the Timeline service from the End Customer, both in terms of handling PII and as a runtime dependency. The Core Platform layer may manage any verification process and ensure that identifiers (e.g. phone number) correctly match stored records in reference to an incoming communication (e.g., a voice call or digital communication).

[0088] Then when the end-customer makes a subsequent communication, the Timeline service receives the end-customer ID and combines historical records from Interactions and (optionally) Topics to form a timeline.

[0089] In example embodiments, profile retrieval and topic updates are managed directly from GUI components on the front-end.

[0090] FIG. **4** is a flow chart showing operations for using the Timeline API to get a Timeline GUI comprising a set of timeline entries.

[0091] In example embodiments, the Timeline GUI is a component provided for surfacing in a GUI, such as a contact center GUI being presented to a representative of an entity for processing interactions associated with an end customer. In example embodiments, data associated with the end customer, such as end customer profile data, is laid out together in the Timeline GUI with communication data associated with historical interactions of the end customer pertaining to an interaction.

[0092] To support the Timeline UI the SDK will handle tokens and communicate with the Timeline Service. The Timeline service is responsible for providing data (e.g., as a RESTful API) for consumption by the SDK, or directly from applications of an entity. It will provide CRUD operations to authenticated consumers. The service will support the following access patterns: custom application "reads" (e.g., to look-up an end-customer timeline), system "writes" to populate the store from internal sources, such as interactions; custom application "writes" to add or modify data in the timeline, including data stored in a timeline data object associated with the timeline.

[0093] The Topics Service may work in two steps. Firstly, an Administrator sets up a topic taxonomy for an organization. Then at runtime, the taxonomy is provided to a Representative who uses it to classify an interaction. For example, Home Loan Application. Using the base taxonomy, a representative can add, remove and group interactions into a logical hierarchy. In example embodiments, the Timeline has a dedicated Timeline store. In alternative embodiments, persistence may be provided through another service, such as the Topics service.

[0094] The Core platform layer is concerned with managing interactions with the end-customer including: handling inbound and outbound communications, transfers, routing communications to users, communication wrap-up, capacity management and invitation handling. The Timeline service receives events from the Interactions service so that date/time and channel information including Channels and Participants can be displayed on the Timeline. Timeline may

depend on the End-customer identifier being present in order to process Topics and Interactions data from the core platform layer.

[0095] Once the End-customer has been verified, the core platform will use the CDP Proxy to ensure that the given identity is mapped to an entry in the CDP, creating a new entry if necessary. The CDP Proxy is primarily responsible for integrating the communication platform with a CDP. The CDP will store End Customer IDs on behalf of the platform. Timeline consumes data from the Customer Identity Service (e.g., using REST).

[0096] FIG. **5** is a block diagram depicting a data flow through an example architecture of the system. In example embodiments, the timeline service processes and shows information corresponding to one or more of the records discussed herein, such as interaction timeline records, including data pertaining to interactions, topics, channels, and participants.

[0097] The Timeline Records are designed to be extensible. For example, Timeline Records can be added as a new type of record to one or more databases. For example, external data from a CDP may correspond to a custom event for adding to the one or more databases.

[0098] In example embodiments, the Timeline service may process PII, but not store it in persistent storage. For example, it may query the PII from origin systems and cache it in volatile memory if needed.

[0099] FIG. **6** is a flow chart depicting an example flow for listing all of the timeline records ordered by created date. In example embodiments, the timeline API is RESTful.

[0100] In example embodiments, the Instance SID identifies an instance with which topics and users are both associated. In example embodiments, the instance may refer to an account, such as an entity account. In example embodiments, an instance may not be required. In example embodiments, a representative may be able to list records across multiple instances, depending, for example, on the representative's role and the access rights assigned to each of the instances. In example embodiments, deleting an instance causes the timeline service to purge the data corresponding to the instance, including the data records associated with the instance.

[0101] FIGS. **7**A-**7**C show an example response to an invoking of the Timeline services API. In example embodiments, the invoking of the Timeline API includes the following request headers (here, the names are exemplary):

[0102] Name: I-Twilio-Auth-Account; type: sid; description: The communication platform account SID.

[0103] Name: I-Twilio-Request-Id; type: sid; description: The communication platform REST API Request SID.

[0104] Additionally, in example embodiments, the invoking of the Timeline API includes the following query parameters:

[0105] Name: pageToken; type: string; description: The token to access the next page.

[0106] Name: pageSize; type: integer; description: The size of the page [1 . . . 1000] (default: 100).

[0107] FIGS. **7**A-**7**C are listings of an example response to a request.

[0108] In example embodiments, schemas for versions, such as for versioning of records, including timeline records, are backward compatible based on their level of volatility. Such schemas having a low level of volatility (e.g., as measured by a configurable threshold value), such as fixed schemas, may be validated (e.g., using a specification, such as the Open API Spec) to make them strongly typed on the contract of the API. The contract may be extensible and/or have an ability to recognize and validate record data.

[0109] Versioning of schemas may be done when the specifications for the records exist and may be able to support a new variable schema design. Schemas may be modeled differently for different types of record types, including external record types and internal record types. External record types, including entity- or user-defined schemas, are not strongly typed (e.g., using Open API), but rather have a flexible schema having metadata in properties for the type of the scheme and the schema version. Internal record types, such as for modeling interactions, voice calls, analytics (e.g., CDP) data, etc. may be strongly typed (e.g., using Open API).

[0110] A role based access control (RBAC) system controls access to the data such that a representative from open account may not be able to see data in another account and vice versa, depending on roles of the representatives and/or their access permissions.

[0111] At the data layer, the partition key will be composed from end-customer+account-sid. The composite key isolates data at rest according to the service instance associated with a communication platform account.

[0112] In example embodiments, any PII, namely: topic names, user (representative) names, customer details, will all be fetched by the UI from relevant services (topics service, user service, customer connect service) based on the sids provided by the timeline.

[0113] Timeline service will read the Account Deleted Events and delete all the data related to that account when that event happens, optional will be create a delete endpoint on account that account janitor can use to delete the data.

[0114] Timeline service will listen to the Instance Deletion event: LifecycleEvent Deletion Initiated Event and delete the data of the instance sid

[0115] The Timeline Builder is a component inside the Timeline Service whose responsibility is to subscribe to events (e.g., Kafka events) and use those events to aggregate data and write to the store.

[0116] The Timeline Builder is designed to handle all write operations against the data store. The implication is that any data that needs to be written in the Timeline Service would need to be consumed as an event.

[0117] FIG. **8** is a flow chart depicting an example of processing interaction events with the Timeline Builder. The list of events to build the entities may include InteractionCreated, InteractionUpdated, InteractionChannelCreated, InteractionChannelUpdate, InteractinoParticipantCreated, and InteractionParticipantUpdated.

[0118] In example embodiments, all records which are obtainable through the timeline API will be partitioned by instance_sid and end_customer_sid (profile_connect_sid). For the events which do not carry this metadata (e.g. channel created event will not carry end_customer_sid/profile_connect_sid), it will be figured out based on the relevant parent interaction. Nevertheless, the system may assume the two scenarios in which case this data will not be immediately available:

[0119] 1. Unidentified Customer

[0120] In example embodiments, in case of the customer not being identified, the system will receive an interaction

created event with no end_customer_sid/profile_connect_sid. In this case the system will store the events partitioned by instance_sid and interaction_sid and will update those upon receiving the InteractionUpdated event once the customer is identified.

[0121] If the customer is not identified during the course of an interaction (e.g., the interaction is closed before the customer is identified), the system may clean up these records as they will not become part of a timeline for any customer/profile.

[0122] Post private beta, we expect events for any unidentified customers to be coming with some anonymous profile sid, in which case we should be able to update those and assign them to the relevant customer/profile even after the interaction is closed.

[0123] 2. Events Coming Out of Order

[0124] We assume that the events could be coming out of order, e.g., the system could receive a InteractionChannelCreated event for a channel being created in an interaction, for which the system did not receive the InteractionCreated event yet. In this case, the system may store the relevant channel record indexed only by the parent record sid (e.g., for channel it will be the interaction_sid) until the system will get the relevant InteractionCreated event, upon which the system may update the metadata.

[0125] FIG. **9** is a table describing an example schema for a timeline record.

[0126] FIG. **10** is a table describing an example schema for interaction data.

[0127] FIG. **11** is a table describing an example schema for a channel.

[0128] FIG. **12** is a table describing an example schema for a participant.

[0129] FIG. **13** is a table describing an example schema for external records. In example embodiments, external records include timeline records generated by an entity using the Timeline service.

[0130] In example embodiments, metadata may be used on the timeline records to label external record types. Inputs may be validated at the service level (e.g., using a schema, such as a JSON schema). The schema may be bound to the API version (e.g., POST/v1/timeline). The schema may be used loosely (e.g., {"minContains": 0, "maxContains": 5} to allow flexibility. A curated list of schemas may be offered (e.g., a Salesforce Schema) for selection by an entity. Each entity may be able to define a schema with some technical checks.

[0131] FIG. **14** is a table of example request headers and an example message body for adding an external record to a timeline.

[0132] FIG. **15** is a sequence diagram showing an example of how an external timeline record may be created.

[0133] FIGS. **16**A-**16**C is a listing of an example entity relationship diagram.

[0134] FIG. **17**A is a listing of getting timeline records with cache.

[0135] FIG. **17**B is a listing of getting timeline records with initialization and events.

[0136] FIG. **17**C is a listing of processing interaction events.

[0137] FIG. **18** is a listing of creating an external timeline event.

[0138] Access Patterns

[0139] In terms of CRUD operations there are three use cases:

[0140] 1. Communication system writes

[0141] 2. Customer reads

[0142] 3. Customer writes

[0143] Communication System Writes

[0144] The Timeline builder component is responsible for ingesting events emitted from the core platform. The platform entities are of the following types: Interactions, Topics, Participants and Channels. Entities are designed to support eventual consistency.

[0145] The most common query is a simple look-up keyed by end-customer and service instance identifiers. This query will be run by Representatives in order to render the timeline in the UI.

[0146] When executing the query, some compensation logic is necessary to compensate for any subsequent writes (updates). For example, removing duplicates and serving only the latest version to the consumer.

[0147] Attributes that are added for querying include: Topics, Date-range and Representative name. Support for paginations and grouping may also be added. The boundary of a timeline record collection may be a single customer. This means that, in example embodiments, all filtering, sorting and grouping operations may only be applied to this result set. In alternative embodiments, searches may scan the entire data store.

[0148] In example embodiments, a Representative\ may o update what is displayed directly from the UI. Furthermore, new records may be added to the timeline by customers using the API directly.

[0149] Records added this way may be called External Records in the model and should be rendered by the UI in near-real-time.

[0150] Data Retention Policy

[0151] In example embodiments, a data retention policy may be implemented (e.g., limiting data retention to two years).

[0152] As historical data approaches end-of-life, entities may be given the opportunity to become the new custodians. During the run-up to the transfer of ownership, they may be invited to use tooling that can analyze and backup the timeline.

[0153] Compliance, Personally Identifiable Information (PII) and Global Data Protection Regulation (GDPR)

[0154] The overall approach is to avoid storing PII in the timeline store, as we are assuming a two-year retention policy. The relevant fields are:

[0155] 1. End-customer name

[0156] 2. Representative name

[0157] 3. External Records

[0158] End-customer name will be resolved by the front-end and only exist as a SID. The Representative name will also be a SID but is more problematic and will require lookups to the User store. To make this efficient a server-side cache may be needed.

[0159] Query User name on each query to the timeline service and put the username in an in memory cache.

[0160] FIG. **19** is a flowchart showing example get timeline events.

[0161] Authorization and Protection

[0162] According to the principle of multi-tenancy a customer cannot access the data of another.

[0163] All timeline records may be stored in one large table that containers for all entities, including all the end-customers. This may be a soft partition as separation between tenants may be logical rather than physical.

[0164] Following the principle of defense in depth, there are four controls to protect unauthorized access.

[0165] 1. Entropy

[0166] 2. Edge Services

[0167] 3. Partition Key

[0168] 4. Encryption

[0169] Entropy means that the mathematical probability of a collision is so small that we can all sleep peacefully. Because the communication platform SIDS have sufficient entropy, end customer SIDS will be universally unique.

[0170] The REST layer provides protection because the Timeline entry is a sub-resource of a service instance. For example /Instances/{SID}/timeline/customers/{SID}. Access to the Instance SID is controlled through the grants allocated during provisioning. Should a representative with an active session attempt to manipulate the URL, the communication platform's RBAC controls will disallow access.

[0171] At the data layer, the partition key will be composed from end-customer+instance. Let's use some pseudo-sql code to illustrate how the control works.

[0172] FIG. **20** is listing of pseudo code of example select statement. In example embodiments, the data at rest is implemented using a key-value store and the pseudo code merely shows logical relationships. The composite key isolates data at rest according to the service instance associated with an account. Finally, data will be protected using table-level encryption and follow a predefined practice for encrypting at rest and key management (AWS KMS). Although client-level encryption would have offered even greater isolation, the overhead of key management doesn't appear to be justified by the business requirement for the data set.

[0173] Data Model

[0174] At a high-level the timeline is a collection of records associated with an end-customer and sorted by time. There are multiple types of records, but initially there will only be Interactions from internal sources.

[0175] FIG. **20** is listing of pseudo code of an example SID.

[0176] Timeline Records Normalized

[0177] FIG. **21** is a block diagram showing a normalized view of an example logical model of the data a timeline may have, including the entity relations. Not this is a logical model, not a physical implementation, such as a SQL schema.

[0178] Timeline Records Denormalized

[0179] Indexes will be used to retrieve and collapse and associate multiple entries to create a single Timeline Record at runtime. For example, DynamoDB supports up to 20 Global Secondary Indexes (GSI) and 5 Local Secondary Indices (LSI). In example embodiments, the system supports multiple (e.g., two) Global Indices. This should support future growth.

[0180] Note about Instance SIDs

[0181] The system's definition may assume that instance_sid is mandatory.

[0182] The implication is that consumers would access the service as follows.

[0183] /Timelines/{endCustomerSid}/Records?flexInstanceSid={flexInstanc eSid}

[0184] In example embodiments, PK could instead be composed with an Account SID and Instance SID would be managed through indexing.

[0185] FIG. **22** is a table showing an example index.

[0186] In example embodiments, all records are partitioned by the account sid and interaction sid PK. There may be no Timeline SID unless there is foreseeable access pattern that would require one. The system may not be using the end customer SID in the partition key because it may be immutable by design.

[0187] When returned to the UI, records are sorted by status identifier first and then by date updated descending. The status identifier is present only for interaction records to fulfil the sorting requirement of showing active interactions first, inactive in the middle and closed in the end.

[0188] Read/Write Pattern:

[0189] Timeline Builder writes data to the database by account_sid and interaction_sid.

[0190] Timeline Service reads the timeline data fetching interactions first by profile_connect_sid using startsWith query on the GSI1. The result is ordered by status_id then by date_modified

[0191] Read timeline details for an Interaction by looking up on the table's primary key. This returns all the items of all types that reference that Interaction. The system may re-order them in-memory by date_modified.

[0192] FIGS. **23-25** are tables showing examples of how data (e.g., interaction, channel, and participant data) will go into a database.

[0193] Version Control

[0194] As well as managing the version of each entity attribute by SID, the schemas that define the entity model may also be managed, such as the main tables the internal entity is source from.

[0195] Each table is defined by a schema in the Avro format. Let's imagine the current versions are as follows:

[0196] {Interactions: 1.1, Topics: 2.0, Participants: 1.0, Channels: 1.0}

[0197] Then in the data store we wish to know which schema version was active at creation time. For example, by binding the above schema versions with an attribute version and storing both in the Sort Key (SK) Field.

Example Response

[0198] Database Selection

[0199] In example embodiments, the following vendors may be considered, but the final selection may be Amazon DynamoDB.

[0200] 1. AWS Aurora MySQL

[0201] 2. Cockroach TDS

[0202] 3. DynamoDB

[0203] 4. ElasticDB

[0204] 5. AWS Aurora MySQL

[0205] 6. AWS Timestream

[0206] 7. Apache Druid

[0207] 8. Clickhouse

[0208] 9. Apache Kudu

[0209] Rationale

[0210] NoSQL may be favoured over SQL because responses may be quick and most use-cases may be simple look-up queries. When writing to the store an Event Sourcing pattern will be used, for which specific databases (e.g.,

DynamoDB) may be a good fit. Generally new entities (interactions) will be appended, although entities that are re-published will be updated.

[0211] Implications

[0212] There is a risk that future requirements will bring more complex queries involving pagination and grouping. Although DDB has more limited querying capabilities than a traditional RDBMS, various tools (e.g., Presto and Hive) may offer the possibility of extending the store to meet those future requirements.

[0213] A NoSQL schema may be different from SQL. For example, DDB is unable to use Table JOINS and has other limitations, for example query results cannot be larger than 1 MB (500 pages of text).

[0214] In order to query topics by interactionSid, a JOIN may be required between Interactions and Topics. In order to retain such relationships, the timeline builder may be configured to transform entities into metadata (GSI indexes) for later retrieval.

[0215] Specific databases, such as DynamoDB, may be the preferred choice; however, SQL may offer more flexibility. For example, something like Cockroach DB would provide support for the GROUP BY function. Developing with DynamoDB does run the risk of rework should it need to be replaced later. The following qualities may help decide:

[0216] HiveQL or Presto enable SQL-like statements on top of Dynamo

[0217] Filtering and Grouping can be achieved with GSIs

[0218] Pagination can be done by multiple queries from the Application Layer (as an exception)

[0219] There is a Select COUNT query in DynamoDB (more info)

[0220] 1 MB is unlikely to be an issue as result sets are limited to a single customer

[0221] Finally, if it becomes necessary to change databases, the system may be configured to support known patterns such, as right-twice, to assist with migration.

Example Mobile Device

[0222] FIG. **25** is a block diagram illustrating a mobile device **4300**, according to an example embodiment. The mobile device **4300** can include a processor **1602**. The processor **1602** can be any of a variety of different types of commercially available processors suitable for mobile devices **4300** (for example, an XScale architecture microprocessor, a Microprocessor without Interlocked Pipeline Stages (MIPS) architecture processor, or another type of processor). A memory **1604**, such as a random access memory (RAM), a Flash memory, or other type of memory, is typically accessible to the processor **1602**. The memory **1604** can be adapted to store an operating system (OS) **1606**, as well as application programs **1608**, such as a mobile location-enabled application that can provide location-based services (LBSs) to a user. The processor **1602** can be coupled, either directly or via appropriate intermediary hardware, to a display **1610** and to one or more input/output (I/O) devices **1612**, such as a keypad, a touch panel sensor, a microphone, and the like. Similarly, in some embodiments, the processor **1602** can be coupled to a transceiver **1614** that interfaces with an antenna **1616**. The transceiver **1614** can be configured to both transmit and receive cellular network signals, wireless data signals, or other types of signals via the antenna **1616**, depending on the nature of the mobile device **4300**. Further, in some configurations, a GPS receiver **1618** can also make use of the antenna **1616** to receive GPS signals.

Modules, Components and Logic

[0223] Certain embodiments are described herein as including logic or a number of components, modules, or mechanisms. Modules may constitute either software modules (e.g., code embodied (1) on a non-transitory machine-readable medium or (2) in a transmission signal) or hardware-implemented modules. A hardware-implemented module is tangible unit capable of performing certain operations and may be configured or arranged in a certain manner. In example embodiments, one or more computer systems (e.g., a standalone, client or server computer system) or one or more processors may be configured by software (e.g., an application or application portion) as a hardware-implemented module that operates to perform certain operations as described herein.

[0224] In various embodiments, a hardware-implemented module may be implemented mechanically or electronically. For example, a hardware-implemented module may comprise dedicated circuitry or logic that is permanently configured (e.g., as a special-purpose processor, such as a field programmable gate array (FPGA) or an application-specific integrated circuit (ASIC)) to perform certain operations. A hardware-implemented module may also comprise programmable logic or circuitry (e.g., as encompassed within a general-purpose processor or other programmable processor) that is temporarily configured by software to perform certain operations. It will be appreciated that the decision to implement a hardware-implemented module mechanically, in dedicated and permanently configured circuitry, or in temporarily configured circuitry (e.g., configured by software) may be driven by cost and time considerations.

[0225] Accordingly, the term "hardware-implemented module" should be understood to encompass a tangible entity, be that an entity that is physically constructed, permanently configured (e.g., hardwired) or temporarily or transitorily configured (e.g., programmed) to operate in a certain manner and/or to perform certain operations described herein. Considering embodiments in which hardware-implemented modules are temporarily configured (e.g., programmed), each of the hardware-implemented modules need not be configured or instantiated at any one instance in time. For example, where the hardware-implemented modules comprise a general-purpose processor configured using software, the general-purpose processor may be configured as respective different hardware-implemented modules at different times. Software may accordingly configure a processor, for example, to constitute a particular hardware-implemented module at one instance of time and to constitute a different hardware-implemented module at a different instance of time.

[0226] Hardware-implemented modules can provide information to, and receive information from, other hardware-implemented modules. Accordingly, the described hardware-implemented modules may be regarded as being communicatively coupled. Where multiple of such hardware-implemented modules exist contemporaneously, communications may be achieved through signal transmission (e.g., over appropriate circuits and buses) that connect the hardware-implemented modules. In embodiments in which multiple hardware-implemented modules are configured or

instantiated at different times, communications between such hardware-implemented modules may be achieved, for example, through the storage and retrieval of information in memory structures to which the multiple hardware-implemented modules have access. For example, one hardware-implemented module may perform an operation, and store the output of that operation in a memory device to which it is communicatively coupled. A further hardware-implemented module may then, at a later time, access the memory device to retrieve and process the stored output. Hardware-implemented modules may also initiate communications with input or output devices, and can operate on a resource (e.g., a collection of information).

[0227] The various operations of example methods described herein may be performed, at least partially, by one or more processors that are temporarily configured (e.g., by software) or permanently configured to perform the relevant operations. Whether temporarily or permanently configured, such processors may constitute processor-implemented modules that operate to perform one or more operations or functions. The modules referred to herein may, in some example embodiments, comprise processor-implemented modules.

[0228] Similarly, the methods described herein may be at least partially processor-implemented. For example, at least some of the operations of a method may be performed by one or more processors or processor-implemented modules. The performance of certain of the operations may be distributed among the one or more processors, not only residing within a single machine, but deployed across a number of machines. In some example embodiments, the processor or processors may be located in a single location (e.g., within a home environment, an office environment or as a server farm), while in other embodiments the processors may be distributed across a number of locations.

[0229] The one or more processors may also operate to support performance of the relevant operations in a "cloud computing" environment or as a "software as a service" (SaaS). For example, at least some of the operations may be performed by a group of computers (as examples of machines including processors), these operations being accessible via a network (e.g., the Internet) and via one or more appropriate interfaces (e.g., Application Programming Interfaces (APIs).)

Electronic Apparatus and System

[0230] Example embodiments may be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Example embodiments may be implemented using a computer program product, e.g., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable medium for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers.

[0231] A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

[0232] In example embodiments, operations may be performed by one or more programmable processors executing a computer program to perform functions by operating on input data and generating output. Method operations can also be performed by, and apparatus of example embodiments may be implemented as, special purpose logic circuitry, e.g., a field programmable gate array (FPGA) or an application-specific integrated circuit (ASIC).

[0233] The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other. In embodiments deploying a programmable computing system, it will be appreciated that both hardware and software architectures merit consideration. Specifically, it will be appreciated that the choice of whether to implement certain functionality in permanently configured hardware (e.g., an ASIC), in temporarily configured hardware (e.g., a combination of software and a programmable processor), or a combination of permanently and temporarily configured hardware may be a design choice. Below are set out hardware (e.g., machine) and software architectures that may be deployed, in various example embodiments.

Example Machine Architecture and Machine-Readable Medium

[0234] FIG. **26** is a block diagram of an example computer system **4400** on which methodologies and operations described herein may be executed, in accordance with an example embodiment. In alternative embodiments, the machine operates as a standalone device or may be connected (e.g., networked) to other machines. In a networked deployment, the machine may operate in the capacity of a server or a client machine in server-client network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a network router, switch or bridge, or any machine capable of executing instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[0235] The example computer system **4400** includes a processor **1702** (e.g., a central processing unit (CPU), a graphics processing unit (GPU) or both), a main memory **1704** and a static memory **1706**, which communicate with each other via a bus **1708**. The computer system **4400** may further include a graphics display unit **1710** (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system **4400** also includes an alphanumeric input device **1712** (e.g., a keyboard or a touch-sensitive display screen), a user interface (UI) navigation device **1714** (e.g., a mouse), a storage unit **1716**, a signal generation device **1718** (e.g., a speaker) and a network interface device **1720**.

Machine-Readable Medium

[0236] The storage unit **1716** includes a machine-readable medium **1722** on which is stored one or more sets of

instructions and data structures (e.g., software) **1724** embodying or utilized by any one or more of the methodologies or functions described herein. The instructions **1724** may also reside, completely or at least partially, within the main memory **1704** and/or within the processor **1702** during execution thereof by the computer system **4400**, the main memory **1704** and the processor **1702** also constituting machine-readable media.

[0237] While the machine-readable medium **1722** is shown in an example embodiment to be a single medium, the term "machine-readable medium" may include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more instructions **1724** or data structures. The term "machine-readable medium" shall also be taken to include any tangible medium that is capable of storing, encoding or carrying instructions (e.g., instructions **1724**) for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present disclosure, or that is capable of storing, encoding or carrying data structures utilized by or associated with such instructions. The term "machine-readable medium" shall accordingly be taken to include, but not be limited to, solid-state memories, and optical and magnetic media. Specific examples of machine-readable media include non-volatile memory, including by way of example semiconductor memory devices, e.g., Erasable Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks.

Transmission Medium

[0238] The instructions **1724** may further be transmitted or received over a communications network **1726** using a transmission medium. The instructions **1724** may be transmitted using the network interface device **1720** and any one of a number of well-known transfer protocols (e.g., HTTP). Examples of communication networks include a local area network ("LAN"), a wide area network ("WAN"), the Internet, mobile telephone networks, Plain Old Telephone Service (POTS) networks, and wireless data networks (e.g., WiFi and WiMax networks). The term "transmission medium" shall be taken to include any intangible medium that is capable of storing, encoding or carrying instructions for execution by the machine, and includes digital or analog communications signals or other intangible media to facilitate communication of such software.

[0239] Although an embodiment has been described with reference to specific example embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the present disclosure. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense. The accompanying drawings that form a part hereof, show by way of illustration, and not of limitation, specific embodiments in which the subject matter may be practiced. The embodiments illustrated are described in sufficient detail to enable those skilled in the art to practice the teachings disclosed herein. Other embodiments may be utilized and derived therefrom, such that structural and logical substitutions and changes may be made without departing from the scope of this disclosure. This Detailed Description, therefore, is not to be taken in a limiting sense, and the scope of various embodiments is defined only by the appended claims, along with the full range of equivalents to which such claims are entitled.

[0240] Although specific embodiments have been illustrated and described herein, it should be appreciated that any arrangement calculated to achieve the same purpose may be substituted for the specific embodiments shown. This disclosure is intended to cover any and all adaptations or variations of various embodiments. Combinations of the above embodiments, and other embodiments not specifically described herein, will be apparent to those of skill in the art upon reviewing the above description.

What is claimed is:

1. A system comprising:

one or more computer processors;

one or more computer memories;

a set of instructions incorporated into the one or more computer memories, the set of instructions configuring the one or more computer processors to perform operations, the operations comprising:

storing data pertaining to a plurality of interactions in a timeline record in real-time when each of the plurality of interactions occurs, each of the plurality of interactions being associated with one or more topics and an identifier of an end customer;

receiving, via an API, a request for a subset of the data that corresponds to an additional interaction of the end customer, the subset identified in one or more parameters of the API; and

in response to the request, providing the subset of the data for surfacing in a user interface such that one or more representatives can make a determination of whether to accept an invitation to process the additional interaction.

2. The system of claim **1**, the operations further comprising:

receiving, via that API, a request for an additional subset of the data that corresponds to the additional interaction of the customer, the request for the additional subset of the data being made in response to a detection that the one or more representatives has accepted the invitation; and

in response to the request for the additional subset, providing the additional subset of the data for surface in an additional user interface such that the one or more representatives can use the additional subset when processing the additional interaction.

3. The system of claim **2**, wherein the additional subset is selected based on one or more topics associated with the additional interaction.

4. The system of claim **2**, wherein the additional subset is selected based on data stored in a profile of the end customer.

5. The system of claim **4**, wherein the data stored in the profile of the end customer is contained in a customer data platform system that is separate from the system and identified by a reference that is added to the timeline record.

6. The system of claim **5**, wherein the reference is added to the timeline record based on an input received from a graphical user interface presented to the one or more representatives.

7. The system of claim **4**, wherein the timeline record is structured to facilitate simultaneous real-time access to the subset of the data and the data stored in the profile of the end customer for presentation in the additional user interface.

8. A method comprising:

storing data pertaining to a plurality of interactions in a timeline record in real-time when each of the plurality of interactions occurs, each of the plurality of interactions being associated with one or more topics and an identifier of an end customer;

receiving, via an API, a request for a subset of the data that corresponds to an additional interaction of the end customer, the subset identified in one or more parameters of the API; and

in response to the request, providing the subset of the data for surfacing in a user interface such that one or more representatives can make a determination of whether to accept an invitation to process the additional interaction.

9. The method of claim **8**, further comprising:

receiving, via that API, a request for an additional subset of the data that corresponds to the additional interaction of the customer, the request for the additional subset of the data being made in response to a detection that the one or more representatives has accepted the invitation; and

in response to the request for the additional subset, providing the additional subset of the data for surface in an additional user interface such that the one or more representatives can use the additional subset when processing the additional interaction.

10. The method of claim **9**, wherein the additional subset is selected based on one or more topics associated with the additional interaction.

11. The method of claim **9**, wherein the additional subset is selected based on data stored in a profile of the end customer.

12. The method of claim **11**, wherein the data stored in the profile of the end customer is contained in a customer data platform system that is separate from the system and identified by a reference that is added to the timeline record.

13. The method of claim **12**, wherein the reference is added to the timeline record based on an input received from a graphical user interface presented to the one or more representatives.

14. The method of claim **12**, wherein the timeline record is structured to facilitate simultaneous real-time access to the subset of the data and the profile of the end customer for presentation in the additional user interface.

15. A non-transitory computer-readable storage medium storing a set of instructions that, when executed by one or more computer processors, causes the one or more computer processors to perform operations, the operations comprising:

storing data pertaining to a plurality of interactions in a timeline record in real-time when each of the plurality of interactions occurs, each of the plurality of interactions being associated with one or more topics and an identifier of an end customer;

receiving, via an API, a request for a subset of the data that corresponds to an additional interaction of the end customer, the subset identified in one or more parameters of the API; and

in response to the request, providing the subset of the data for surfacing in a user interface such that one or more representatives can make a determination of whether to accept an invitation to process the additional interaction.

16. The non-transitory computer-readable storage medium of claim **15**, the operations further comprising:

receiving, via that API, a request for an additional subset of the data that corresponds to the additional interaction of the customer, the request for the additional subset of the data being made in response to a detection that the one or more representatives has accepted the invitation; and

in response to the request for the additional subset, providing the additional subset of the data for surface in an additional user interface such that the one or more representatives can use the additional subset when processing the additional interaction.

17. The non-transitory computer-readable storage medium of claim **16**, wherein the additional subset is selected based on one or more topics associated with the additional interaction.

18. The non-transitory computer-readable storage medium of claim **16**, wherein the additional subset is selected based on data stored in a profile of the end customer.

19. The non-transitory computer-readable storage medium of claim **17**, wherein the data stored in the profile of the end customer is contained in a customer data platform system that is separate from the system and identified by a reference that is added to the timeline record.

20. The non-transitory computer-readable storage medium of claim **19**, wherein the reference is added to the timeline record based on an input received from a graphical user interface presented to the one or more representatives.

* * * * *