Helm: Self-Service Customer Data Platform

A Technical Paper prepared for SCTE by

Sriharsha Gangam
Principal Architect
Comcast Cable
1800 Arch St, Philadelphia, PA 19103
+1 215 583 8078
sriharsha_gangam@comcast.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

For any organization or business, the importance of good customer experience cannot be overstated. Capturing and understanding customer interactions with the business is the first step towards empathizing with customers. For large organizations and conglomerates with millions of customers and multiple domains or business areas, typically the data is scattered across several business units. It becomes challenging to understand the holistic customer experience. For example, the data from domains such as billing, customer care, audit, and operational interactions could be generated by siloed business units. To understand customer experiences, it is important have these data points stitched together -- particularly when there are multiple products and services available to a customer.

These challenges are industry-agnostic and apply to industries beyond cable and telecommunications. For example, in the retail industry, it is important to understand customer interactions for products and services across multiple channels (e.g., marketing, orders, billing, shipment and call-center interactions). A customer might purchase certain merchandise online, exchange it with another product at a retail store, and subscribe to a new monthly payment installment plan. It is expected that all these interactions are captured and linked together to serve and assist customers. Similarly, in healthcare, it becomes important to have a comprehensive view of health records for all patient interactions, even as the health data could originate from several independent health care providers.

To address these goals, organizations build customer data platforms (CDP). A CDP consolidates and integrates customer interaction data from multiple domains to build and present a unified profile around each customer. At the heart of every CDP is a metadata management system to enable ingestion, governance, and access to the data. Once the data is available, it enables one to gain insights into customer experiences, troubleshoot problems, and interact with customers to improve their experiences.

While it is important to be data-centric, the dynamic nature and increasing variety of customer interaction datasets introduces new challenges. It is ever more important to manage customer data in a safe, secure, and self-service manner. While rich applications can be built on a CDP, data governance and ownership responsibilities can be overwhelming if they are not democratized and decentralized.

In this paper, we will present Helm, which is the name of one of Comcast's CDPs that manages data ingestion, processing, and consumption of anonymized customer data for purposes of advanced issue resolution. We share our experiences to understand what it takes to effectively build and maintain a CDP. The Helm platform ingests hundreds of event datasets from multiple domains, such as customer care, audit, billing, device activations and operations. To put the scale in context, the platform captures tens of billions of customer interactions from millions of customers every month. Internally, it is directly used by tens of thousands of employees to serve several hundred thousand customers, every month. An important component, Helm application programming interfaces (API), play a crucial role across several products in Comcast, serving several billions of API requests every month.

The paper is organized as follows: we will begin by introducing some of the challenges in building, maintaining, and governing CDPs. We then present an overview of the Helm CDP and its impact on customer experiences. This is followed by the platform architecture and design principles. We conclude the paper with a look at Helm's journey so far, and possible future states.

2. Helm Overview

Everything discussed in this paper as “Helm” dates back to the 2012 timeframe, when a group of Comcast employees developed a “Lab Week” project, called “Timeline,” to help us to better stay in touch with our customers who experience service issues. During its conception and the early stages, Helm was composed of bespoke data engineering extract, transform and load (ETL) steps. Each dataset had separate plumbing (data pipelines and code) to extract data from external database systems and apply custom business logic to transform the data. This was hard to maintain, as the data evolved over time. The Helm organization was responsible for data ownership, management, and governance of these datasets. The lack of domain expertise made it challenging for consumers to leverage the data. As the number of datasets grew in terms of domains and variety, data onboarding was hard to scale up, due to limited team capacities. This approach became untenable.

Helm Self-Service was built to address these problems. It represents the control plane or management layer of the Helm CDP. With it, data originating teams can define metadata and configurations about the datasets they publish into Helm. Data producer teams tend to be data business owners, data managers, and domain experts; the platform enables a seamless communication between the domain experts and data consumers. Dataset onboarding and governance deliberately do not require engineering effort by the Helm team. Instead, they are consistent, decentralized and democratized across business domains. As of this writing, a few hundred datasets are onboarded and made available in Helm. Several datasets are modified or added every week. Helm Self-Service user interface (UI) and APIs made this possible.

The Helm team has been in pursuit of continuously improving the Helm Self-Service user experience. Helm is integrated with Matomo (*https://matomo.org/*), InMoment (*https://inmoment.com/*), and Intercom (*https://www.intercom.com/*) to capture usage metrics and product net promoter score (NPS). Data onboarding and metadata management can have several configuration steps and occasionally require additional support from Helm administrators. The Helm success team leverages tools such as Atlassian (*https://www.atlassian.com/*) service desk, Slack (*https://www.slack.com/*) and Intercom for assisting Helm users.

Data's value is only as good as its quality. With Self-Service, data quality control responsibilities are shared with data producers. Helm is integrated with DataDog (*https://www.datadoghq.com/*) to expose data and platform observability metrics. Data availability (e.g., missing data) and quality (e.g., unexpected, or missing attributes) metrics are captured, summarized, and saved for historical lookups. These metrics are sent back to the data owners to resolve any data quality and availability issues. In addition to data quality, a metadata completeness score is captured to ensure that the metadata (e.g., attribute level documentation, tags) is up-to-date and accurate (see Figure 1). This way, Helm users can browse, understand, and consume data correctly as it evolves. Helm as a CDP platform publishes its own platform metrics, logs, and traces into DataDog to enable operational support and expose Helm's service level objectives (SLO).

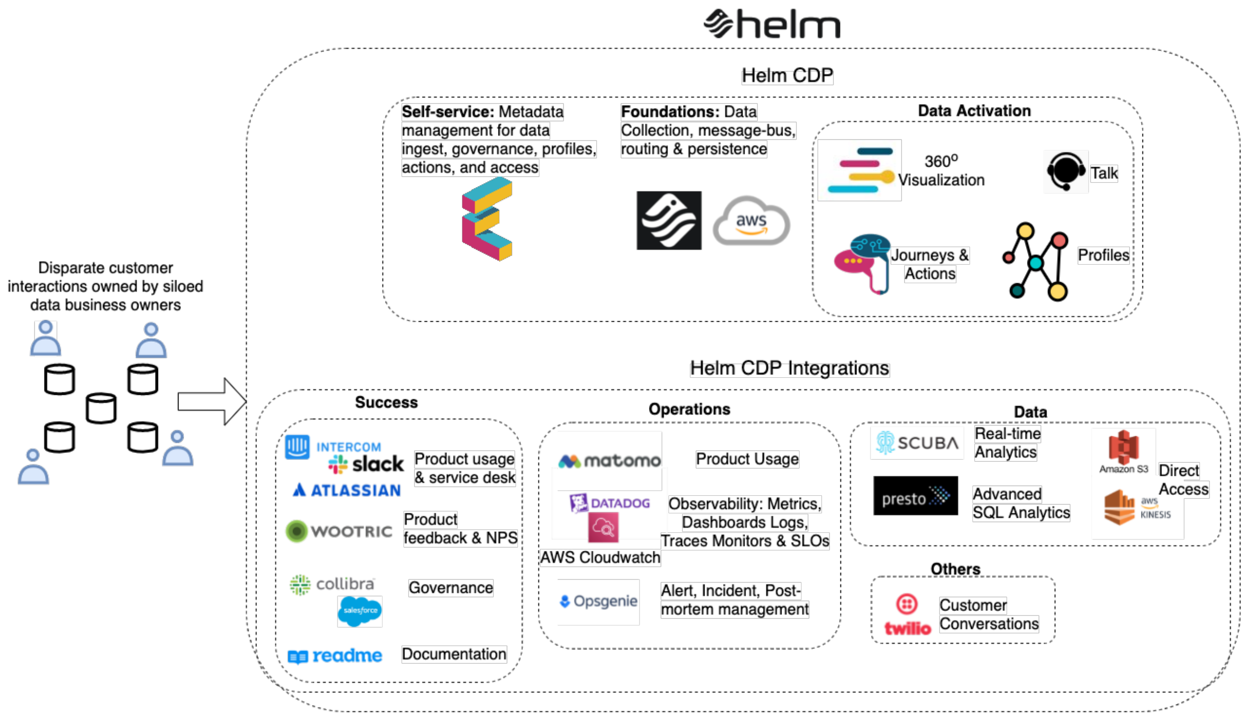


Figure 1 – Helm Overview

Once high-fidelity customer data is available, the next step is to enhance customer experiences and bring business value. Helm Applications is collection of packaged services that is included and integrated with the Helm CDP. The applications are a 360º Visualization application (timeseries visualization), Talk (customer conversations), and Analytics. They are designed to work together, draw out valuable data insights, and enable users (employees) to deliver the best customer experiences. To enable and support a variety of applications, data organization is foundational for CDPs. The next section introduces the Helm data model, which describes how data is stored within Helm. This is followed by an overview of Helm Self-Service for ingesting, managing, and governing the data.

Data Model

A profile (or entity) is the principal or thing around which timeseries event data is captured. For example, a "customer" profile is associated with a variety of customer interaction timeseries datasets such as product activations, billing data, or troubleshooting tickets. Such timeseries datasets could be originating in different business units within an organization. Each entry in these event datasets would have a unique customer identifier (the profile identifier) to link back to the customer profile, a timestamp to understand event chronology, and other attributes relevant to the specific dataset type. A product activation dataset could have additional attributes to capture a product activation experiences (e.g., attributes about the type product or service, lead time to activate the product, and activation failures).

The Helm data model supports different types profiles that may or may not have relationships with each other (see Figure 2). In addition to the customer profile, the "account" and "device" profiles are relevant in the cable and telecommunications domain. Every profile has a unique identifier and certain attributes. A customer's profile could have essential attributes to lookup, verify and assist a customer such as the name and contact information. An account profile could have attributes such the account activation date, service type, and a unique account number. If customers can open multiple accounts, this is represented by a one-to-many relationship between a customer profile and each opened account. Certain timeseries event datasets, such as truck roll schedules, may apply to accounts rather than customers. For each account, multiple devices could be registered that generate device availability events. These datasets directly apply to devices (not customers or accounts). It is important to define and maintain profile relationships (*i.e.,* relationships between customers, accounts, and devices) to understand, navigate, and correlate events across datasets. These profile relationships are foundational in Helm and are represented by a graph – the profile graph. The profile graph along with timeseries datasets for each profile type constitute the Helm data model.

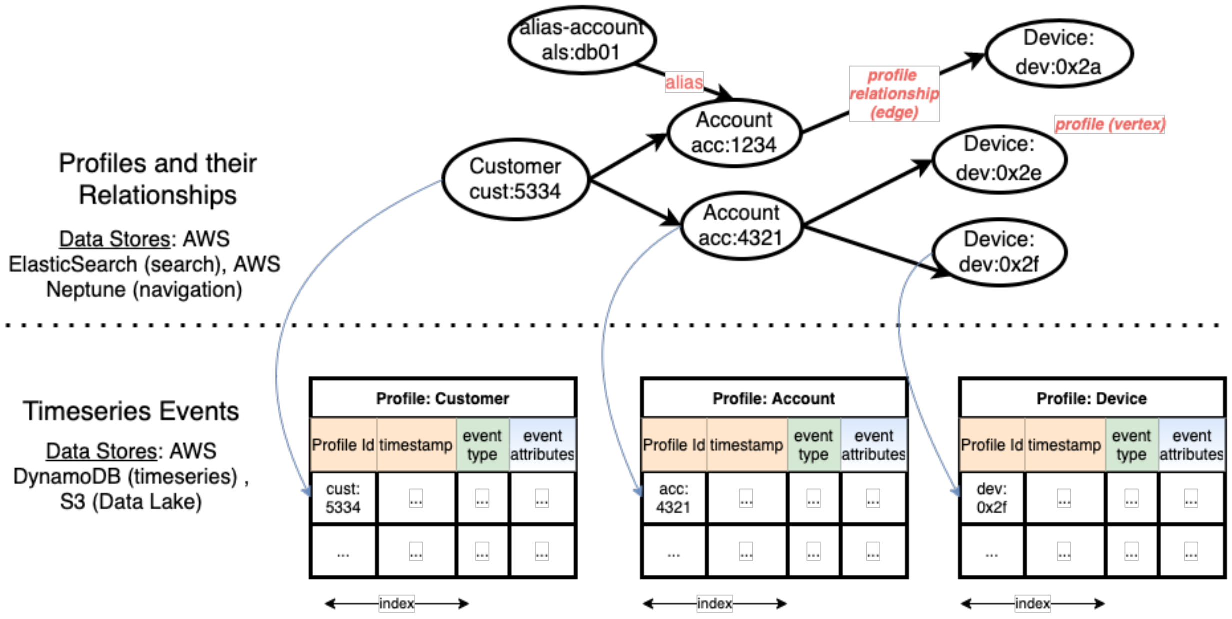


Figure 2 – Example Helm Data Model

2.1. Self-Service

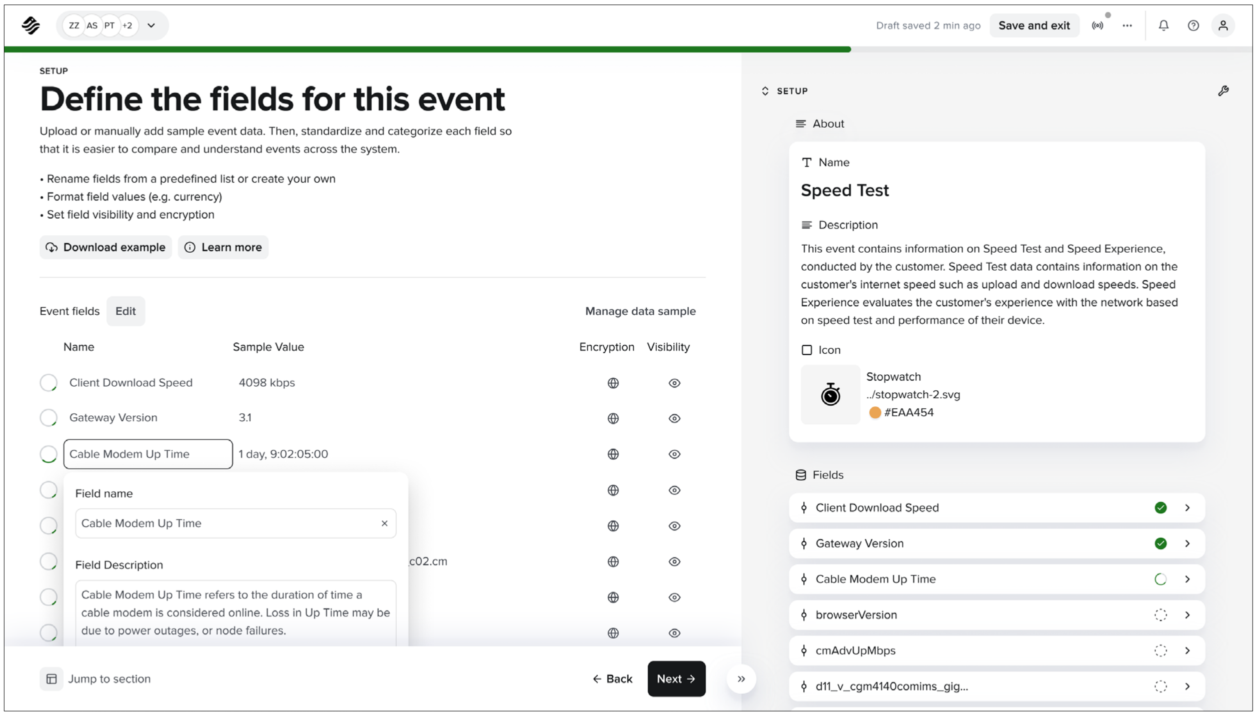
First and foremost, a CDP needs to provide services to capture data effortlessly. Data owners across the organization should be able to publish data into Helm with minimal supervision. Data owners use Helm's Self-Service portal to define schema and other onboarding metadata. While the dataset schema is determined by the data producer and the business context, it must adhere to Helm's data model. It must have a timestamp (for timeseries data), a profile identifier, and additional attributes. Once a dataset is onboarded, the Helm platform will orchestrate the creation of specific infrastructure to enable data ingestion. Orchestration creates configurations to define and manage the data plane. For example, a dedicated hypertext transfer protocol secure (HTTPS) API endpoint is made available for the accepting the data. Additional application-specific orchestrations are executed when necessary.

As stated previously, Helm Self-Service represents the control plane or management layer of Helm CDP. In addition to incoming schema metadata, it enables metadata capture of transformations, routing, persistence, lifecycle, and access controls. The data producers configure the following metadata while onboarding a dataset (See figure 3):

Self-Service Metadata – Control Plane

- **Source Metadata**: Includes all the metadata relevant to a source dataset; e.g., security credentials, feed schema, and documentation. Security credentials ensure that only authorized users and applications produce data into the Helm platform. Data onboarding requires producers to configure necessary credentials upfront. For real-time data ingest, Helm supports client authentication and authorization using OpenID connect (OIDC) and open authorization, version 2.0 (OAuth 2.0). The feed schema defines the structure of data published into Helm. It includes expected attribute names [JavaScript object notation (JSON) paths], expected data types, and expected values as per JSON schema specifications. The feed schema is used to validate ingested data and errors are communicated back to the data producer. The data producer will either need to fix the schema or the data being published. Every feed schema is accompanied by documentation that describes the dataset and attribute definitions with necessary business context. Documentation is important to build good data dictionaries. Helm users can search the dictionary to identify datasets of interest.

- **Transformation Metadata**: Includes all metadata definitions to transform the ingested data; e.g., normalization and encryption schemas. The encryption schema identifies sensitive attributes that require encryption. Helm encrypts these attributes while ingesting the data. Decryption of these attributes is restricted to privileged users and applications. The normalization schema defines naming and structural transformation specifications. It determines a target data schema that is persisted in Helm. Often times, data owners would like to change the name of an attribute or the structure of data stored in Helm without depending on the source dataset. This decouples data production from consumption and provides flexibility when data producers are unable to update their schemas. However, historical data processing can become tedious if the attribute names are updated over time. For example, analytical SQL queries would require multi-part queries, one part for each attribute name. Instead, the datasets flowing into the Helm data lake (for analytics use cases) are transformed. The transformation reverts attributes to their oldest (unique) registered names. This ensures consistent attribute names across historical data.
- **Routing, Persistence and Access Metadata**: The routing metadata for a dataset determines if one or more Helm applications (such as 360º Visualization, Analytics, or watchlists) are subscribed to the dataset. Not all datasets are suitable for every application and use case. The access metadata defines fine-grained access control policies for datasets and attributes based on their sensitivity levels. Persistence metadata defines the data lifecycle and retention goals for the dataset. These are influenced by an organization's data compliance and governance policies. Retention and lifecycle management features of underlying data stores are leveraged when applicable.
- **Application Metadata**: Helm is integrated with applications such as 360º Visualization, Talk, Watchlist, Profile Search & Navigation, and Analytics. Self-Service extends to these applications and enables users to configure them. For example, users can define icons for every data type in 360º Visualization. The profile schema metadata is applicable when the dataset is used to build a profile graph or used to define profile translations. It contains profile attributes, identifiers, and relationships between profile types. It also determines keyword-searchable attributes for a profile.



SETUP

Define the fields for this event

Upload or manually add sample event data. Then, standardize and categorize each field so that it is easier to compare and understand events across the system.

- Rename fields from a predefined list or create your own
- Format field values (e.g. currency)
- Set field visibility and encryption

Name	Sample Value	Encryption	Visibility
Client Download Speed	4098 kbps		
Gateway Version	3.1		
Cable Modem Up Time	1 day, 9:02:05:00		
	c02.cm		

Field name: Cable Modem Up Time

Field Description: Cable Modem Up Time refers to the duration of time a cable modem is considered online. Loss in Up Time may be due to power outages, or node failures.

Name: Speed Test

Description: This event contains information on Speed Test and Speed Experience, conducted by the customer. Speed Test data contains information on the customer's internet speed such as upload and download speeds. Speed Experience evaluates the customer's experience with the network based on speed test and performance of their device.

Icon: Stopwatch ../stopwatch-2.svg #EAA454

Fields
- Client Download Speed ✓
- Gateway Version ✓
- Cable Modem Up Time
- browserVersion
- cmAdvUpMbps
- d11_v_cgm4140comims_gig...

Figure 3 – Helm Self Service Data Onboarding

Ingestion & Persistence – Data Plane

After a dataset is registered in Helm Self-Service, the producer can publish the data into Helm. For ingesting real-time data into Helm, a dedicated HTTPS endpoint is created for each onboarded dataset. In this scenario, the ingested data is available for consumption in near real-time. For batch use cases (e.g., data is produced hourly or daily), data producers can directly push files to a dedicated Amazon Web Services (AWS) S3 bucket using temporary credentials generated by Helm's API. Unlike real-time ingestion, batch ingestion delays are less predictable due to their underlying nature (e.g., asynchronous and throughput-optimized).

For each event datum received from the producer, Helm performs a series of validation steps as per the security credentials and feed schema defined in Self-Service. The datum is then transformed in a sequence of steps such as normalization and encryption (if necessary). If any of these steps fail, the datum is rejected, and the data producer is notified. Helm provides live and historical observability of failures and rejected data. If all the steps execute successfully, the datum is accepted and is written to Helm's message bus for routing and consumption.

A datum's journey from the message bus is determined by the routing metadata configured in Self-Service. Helm applications consume this datum and may perform certain compute operations (e.g., sessionize, evaluate rules, aggregate, enrich, or store) specific to the application. Finally, the datum is persisted in datastores appropriate for the query use case. To enable profile keyword search and navigation based on profile attributes and relationships, the datum is stored in a modern graph database (AWS Neptune) and search database (AWS Elasticsearch). Similarly, to support real-time event lookups (timeseries data) for a profile, the datum is stored in a document store (AWS DynamoDB). For long term trends and aggregate analysis, the data is batched and written into the Helm data lake (AWS S3 and AWS Glue).

3. Data Activations

In the previous sections, we provided an overview of the Helm CDP and the importance of Self-Service in scaling the platform, and the underlying data model. In this section, we describe how employees engage with customers to enhance customer experience using Helm applications like Profile Search & Navigation, 360º Visualization, Talk, Journeys & Actions, and Analytics. Datasets that are onboarded once are reusable across these applications. This section describes how these applications complement each other and provide value that is more than the sum of its parts. Helm datasets are made available to non-Helm applications via data integrations and APIs.

3.1. Profile Search and Navigation

Once all the relevant datasets are onboarded and ingested into a CDP securely, the platform provides capabilities to query and serve this data. Helm's Profile Search & Navigation helps employees use search keywords to identify customers. For example, a care agent could use a customer's street address or a device's media access control (MAC) address to identify a customer profile. After identifying the correct profile, the account and device timeseries event data can be explored using 360º Visualization. A hypertext markup language (HTML) form accepts a keyword and returns all profiles that match the keyword. Auto-suggest results are provided for every keystroke, to enhance the search experience. The profile search application is served by APIs for programmatic integrations. The API accepts keyword requests to match with searchable profile attributes and returns a collection of matched profiles. It also accepts a profile identifier and allows one to navigate profile relationships and traverse to the profile graph.

The Profile API also aids in data ingestion, such as when a dataset does not have expected profile identifiers. For example, Comcast has several biller integrations. Depending on the services, customers could have multiple accounts, one for each biller. To serve customers effectively, it is important to transform biller-specific datasets to a common account identifier. Using profile relationships between biller accounts, the ingestion pipeline can transform the account identifiers of a dataset.

3.2. 360º Visualization

360º Visualization provides an intuitive and exploratory experience for understanding customer (profile) interactions. Customer interactions for a given customer's globally unique identifier (GUID) (i.e., profile identifier) occurring within a time range are presented in a single, chronological view. This "lifetime view" enables employees to serve customers more effectively, because they can see, sequentially, all current and previous interactions, without having to access disparate databases and systems. An employee can understand the customer's journey with historical data, as well as serve customers with real-time status and information. The visualization enables employees to seamlessly drill-down to a specific customer event of interest, from what can be many interactions associated with that customer. Figure 4 shows a screenshot of 360º Visualization, where events are represented by icons. In practice, clicking on these icons expands the event with necessary business context. The range-selector box (black) at the top of the screen helps users choose the timeframe of interest. The "swim lanes" and event filter selections enable one to hide irrelevant events. Visualization user preferences can be customized and saved.

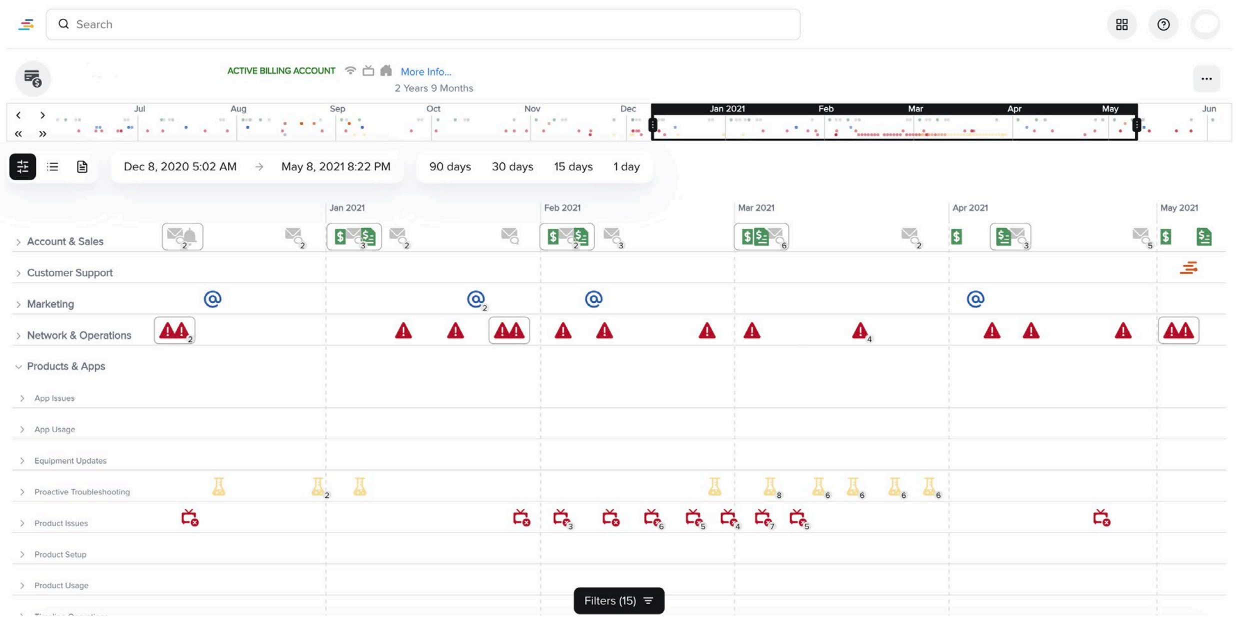


Figure 4 – 360° Visualization - Chronological View of Customer Interactions

Behind the scenes, the UI is backed by the Helm Events API. The Events API accepts a profile identifier, profile type, a collection of event types (optional), and a timestamp range (optional) and returns all events occurring within the time frame. These events are fetched from Helm's timeseries document store that is indexed on profile identifiers and timestamps. The Events API integrates and serves several critical Comcast applications external to Helm. At the time this paper was written, the Events API serves several billion API calls monthly, and the 360° Visualization UI serves several hundred thousand visits every month.

3.3. Talk

As employees troubleshoot customer concerns in 360° Visualization, they might want to reach out to the customer. Helm is integrated with "Talk", a collection of services that initiate short message service (SMS) messages, email, chat, or phone conversations with customers. This contact information is made available from Helm Profiles. No matter who or how many care agents communicate with a customer, all customer conversations are captured as events and ingested back into Helm. The 360° Visualization application allows one to review all these conversations in context of a customer's overall experience. Conversations with a shared context across channels and care agents are captured in a single place. Conversations are seamless, and care agents assisting the customer are literally all on the same page.

Helm Talk has played an important role with the NPS callback program at Comcast. Customer profiles with unsatisfactory NPS surveys are made available to certain employees (via Helm Analytics). The employees are required to proactively reach out to these customers to understand and address their concerns. With the integrated Helm Talk and 360° Visualization capabilities, employees can effortlessly review and initiate customer conversations.

Behind the scenes, Helm integrates with Twilio APIs (*https://www.twilio.com/*) to provide these communication capabilities. During the Covid-19 pandemic, Helm Talk capabilities were expanded to support video calls. This enabled our workforce to extend troubleshooting capabilities with audio and video communication without entering customer homes. This success story was feature in Comcast's corporate blog (*https://corporate.comcast.com/stories/digital-solutions-keeping-us-connected-covid-19*).

3.4. Journeys & Actions

Once customer troubleshooting is completed via the 360º Visualization and Talk applications, a care agent might wish to get notifications (triggered by specific events) about a customer, to ensure that there are no chronic issues. Similarly, care agents and customers might want to get notifications for planned network outages occurring within a certain neighborhood. These examples require configuration of event-driven rules and actions.

An event-driven rule defines the business criteria when a certain action (e.g., a customer notification) is performed. In the above example, the rule evaluation determines if an account belongs to a specific neighborhood and if the interaction type is a network outage. Information to evaluate a rule is typically available as attributes within the event and is enriched by external systems if necessary. In theory, actions could be defined to trigger any external service API call making them extensible.

The above examples do not consider a customer's historical events or prior context for evaluating event-driven rules. For example, a care agent might be interested in truck roll notifications that are re-scheduled more than twice. This requires information about a customer's journey or historical state to evaluate rules and execute actions. A "journey," in this context, is a snapshot of a customer's experience, built from a collection of seemingly disparate or distinct events occurring within a time period. A customer's truck roll journey state machine, for instance, could transition between the following states: Created, scheduled, completed, cancelled, or re-scheduled (see Figure 5). As individual events (e.g., "created," "scheduled") are observed, the workflow progresses, and the journey state is transitioned to a next one.

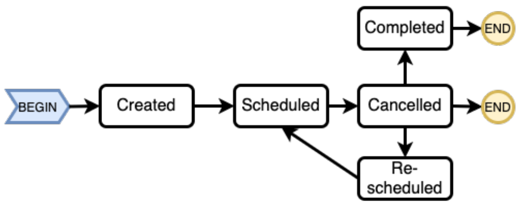


Figure 5 – An Example Truck Roll Journey

In 360º Visualization, care agents would need to mentally re-create a customer's journey by looking at individual historical events to understand the overall context. For complex workflows, this incurs a heavy cognitive load, every time. A journey state machine visualization aids this understanding and presents a customer's journey that could be understood with a single glance. The journey state machine is defined by business process flows and managed with workflow engines such as Flowable (*https://flowable.com/*). Each state transition creates an event in Helm representing the source and target states. By defining event-driven rules on journey state transition events, complex business rules and actions can be accommodated. Rule evaluation remains a stateless operation and is decoupled from journey state management.

Helm is integrated with 'Watchlist', an application that lets employees get notifications about customer events. This is achieved with a few clicks from the 360º Visualization experience. For example, a watchlist rule could be - "For new network outage events matching account numbers in the list [0003, 0004, 0005], trigger an SMS & email notification". The subscribed employees receive watchlist notifications via email or text (the action) along with the customer and event details that triggered the notification.

3.5. Analytics

It is valuable to review a single customer's experience in 360º Visualization or Journeys, but this can be hard to scale. With millions of customers, it becomes important to understand aggregated customer experiences across the population. This is accomplished with Helm Analytics, which provides both business insights and analytics capabilities. For example, it is useful to understand the population distributions at each step of a business workflow journey. This can be visualized with Sankey diagrams and infographics as shown in Figure 6. This helps business teams identify common problems across customers, and invest in areas that provide the most value. Product owners can understand the impact of their feature rollouts on customer journeys by correlating with product NPS. This fast feedback enables organizations to adapt and innovate with short lead times. Similar arguments apply for initiatives that aim to reduce an organization's budget expenses without impacting the customer experience.

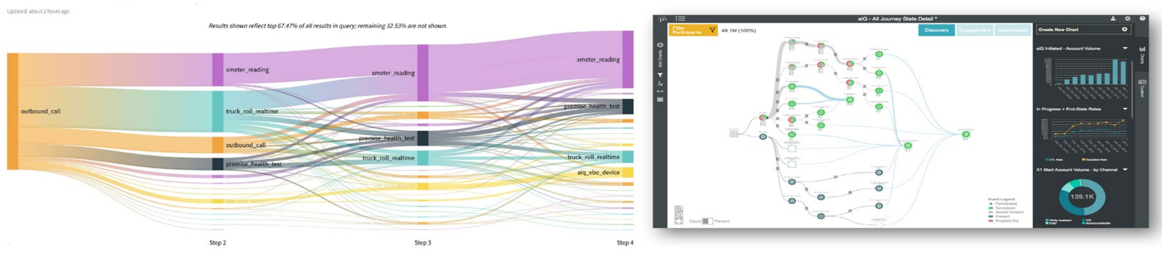


Figure 6 – Journey Analytics from Scuba and Pointillist

The NPS callback program at Comcast lets certain employees query Helm Analytics for customers with a low NPS. With a list of such customers, employees could review events in 360º Visualization, follow up with Talk, and query for customers with similar problems. These complementary applications provide the flexibility to switch context from an aggregate population view to individual customers and vice-versa, and demonstrate the immense value of Helm CDP at Comcast.

Datasets within Helm are processed and made available for analytical queries using modern data stores and analytical query engines. Users are exposed with a UI & API for interfacing with the underlying query engine. Like 360º Visualization, data access is managed using user groups and roles. At Comcast, Helm is integrated with Scuba (*https://www.scuba.io/*) and Pointillist (*https://www.pointillist.com/*) to provide near real-time analytical query capabilities. The platform serves tens of thousands of queries every month from thousands of users. The underlying datasets are partitioned and managed in Helm's data lake (backed by AWS S3andAWS Glue) to enable integration with other analytics products.

4. Helm Architecture & Design

In the previous sections, we provided an overview of Helm and showed how the applications activate data to improve customer experiences. In this section, we will present the building blocks of the Helm platform. This section details the design principles guiding the architecture, an architectural overview, and the platform's evolution over the years.

4.1. Design Principles

Below are some of the design principles that helped shape Helm's current architecture.

1. ***Metadata Driven, Domain Agnostic, and Multi-Tenant Capable***: Configurability was an essential architectural consideration, because it enables the re-use of functionality or services. Self-Service is intended to manage all metadata configurations per dataset. Observability metrics

are provided to understand the impact of configuration changes on the datasets. Platform configurability enables Helm to be domain- and industry-agnostic, and, as a result, to support multiple business verticals and industries. Profile types, their relationships, and associated time series datasets can be defined for each domain. Branding and user experiences (e.g., 360^o Visualization swim-lanes) are configurable. Organization-specific integrations could be built with Helm SPIs (service provider interface), and as such support multi-tenancy by providing dedicated Helm instances in tenant namespaces. A Helm instance is created by cloning the infrastructure in a tenant namespace and applying tenant configurations. Helm is in the process of transforming into a shared-services, Software-as-a-Service (SaaS) model to reduce the time necessary to onboard new tenants.

2. ***Near Real-time Streaming with High-Fidelity***: Helm is designed for near real-time data ingestion and stream processing applications. Delays are unacceptable, from a usability standpoint, in applications like 360^o Visualization and Watchlist (for troubleshooting and notifications). Certain other applications (e.g., Analytics) may be more delay-tolerant, especially if data efficiency is the priority, in which case data can be processed in batches. Data validation errors are immediately made available to the data producers. This enables data fidelity, creates a fast feedback loop with the data producer, and promotes dataset evolution.

3. ***Cloud Infrastructure and Performance Isolation***: Helm is designed to leverage cloud efficiencies, having been built using several managed and serverless AWS technologies (e.g., Beanstalk, Lambda, Kinesis, Firehose, S3, IAM, KMS, DynamoDB, Elasticsearch, Neptune, and RDS). Use of the public cloud as a technology choice is influenced by factors including corporate technology guidelines, Helm's business and strategic objectives, and personnel budgets.
Helm is designed to provide isolation and performance guarantees per dataset. For example, infrastructure orchestration (via Self-Service) creates a dedicated stream (AWS Kinesis) and compute infrastructure (AWS Lambda) for each dataset. The downside is the additional overhead for managing this infrastructure. The trade-off could be tuned by enabling shared infrastructure configurations. For example, a message bus with 's' streams could share 'd' datasets, where $1 \leq s \leq d$.

4. ***Delivery and Access Guarantees***: The Helm ingestion pipeline provides "at least once" delivery guarantee. "Exactly once" delivery requires every component in the data pipeline to guarantee it, and is generally hard to achieve. Assuming it is achievable, a data producer might still publish duplicate data into Helm. Instead, Helm provides "at least once" delivery guarantees and requires consumer applications to be idempotent. For 360^o Visualization, the data store overwrites an event record if it receives another datum with the same idempotency key. The idempotency key uniquely identifies an event and is generated from a strong one-way hash function of event datum attributes. Similarly, for applications such as Journeys and Actions, the underlying state machines are designed to be idempotent. All aspects of Helm access, the product UIs, APIs, data, AWS services, and third-party integrations are secured by access controls with the principle of least privilege. All data is encrypted in transit and at rest. Additionally, sensitive attributes are encrypted at the application layer. Only authorized users and applications can decrypt these attributes.

4.2. Architecture Overview

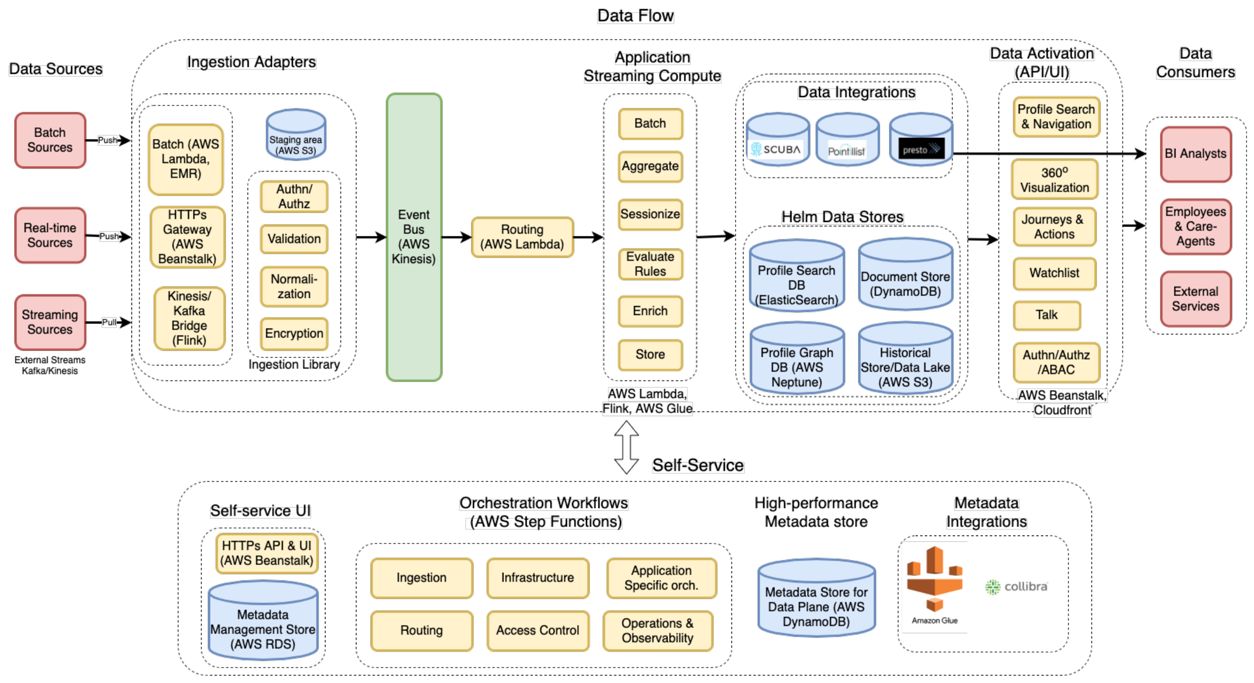


Figure 7 – Helm Architecture

Helm's high-level architecture, illustrated in Figure 7, starts with the foundational Self-Service control plane for metadata management. Self-Service is intended to manage datasets for all aspects of the data flow, from ingestion into Helm to consumption from Helm. Dataset registration in Self-Service triggers a workflow of orchestrations to enable ingestion, routing, infrastructure, persistence, access, and observability.

Ingestion adapters receive data from real-time, batch, and streaming sources. Data ingestion includes a sequence of steps such as authorization, normalization, encryption, and publishing to the Helm message bus. These steps are implemented as shared libraries and are reusable across different compute engines (AWS Beanstalk, EMR, Flink). A datum is said to be accepted in Helm only after it is written to the message bus.

Data from the message bus is consumed and written to Helm's historical data store (AWS S3). A dataset's routing configuration determines if it is consumed by one or more Helm applications. These applications and data integrations (e.g., Scuba, Pointillist) process, persist, and provide access to the data (via APIs and UIs). Applications consume the datasets independently using streaming compute (e.g., they transform, aggregate, batch, evaluate rules, sessionize, enrich, or store). For example, certain timeseries datasets are sessionized to reduce noise in 360° Visualization before persisting. For 'Watchlist', matching rules are evaluated to trigger notifications. Profile datasets are transformed into vertex and edge representations for storing in a graph database.

Occasionally, there are no data producers to publish profile datasets into Helm. In such cases, timeseries event data is processed to identify profiles missing in Helm. Profile-specific SPIs are used to fetch these profiles from source systems and ingest into Helm. These SPIs are also used to fetch profile updates from source systems. Similarly, certain timeseries datasets do not have data producers. Helm maintains a collection of scheduled jobs to periodically fetch these datasets from source systems and ingest them into Helm.

4.3. Helm Evolution - Past and Future

Helm was initially conceptualized to support 360º Visualization. Back then, the platform was simply called "Customer Timeline". The Timeline web-application was originally built on Comcast's cloud infrastructure. It was designed to capture and visualize customer interaction (timeseries) data. Dataset onboarding required engineering new data pipelines to fetch data from source systems. It was time consuming to onboard new datasets. The 'Timeline API' enabled data access for external applications. Around this time, initiatives for migrating on-premise services to AWS public cloud had begun. Self-Service was introduced to catalog and manage datasets. Self-Service enabled dataset orchestrations and auto-created data pipeline infrastructure. Self-Service enabled the platform to scale using a shared responsibility model for datasets. However, day-to-day data operations were time consuming as the number of datasets scaled. Data producers could not get immediate feedback on the data quality. Customized dashboards were built for each dataset to monitor volume anomalies.

Helm was built to better understand customer journeys across every product and channel where they interact with Comcast. Data visualization tools and data APIs for sharing customer context with other applications are the cornerstone capabilities that enabled Helm to scale throughout Comcast. As high value datasets were onboarded, other applications such as Watchlist, Talk, Analytics, and messaging applications were introduced to leverage this data. Again, these applications had bespoke data pipes consuming from the Timeline's message bus. These applications required engineering work for every new onboarded dataset. Platform investments expanded Self-Service to support analytics and other applications. Dataset orchestrations that were originally intended for ingestion were expanded to support routing datasets to applications. At the same time, the applications evolved to consume these datasets and provide rich experiences. Eventually, the applications were integrated and managed with Self-Service and the platform was rebranded to Helm in 2019. Enhanced observability and process improvements enabled data producers to address the data quality and volume anomalies.

Over time, Self-Service and orchestrations expanded in complexity. Orchestrations were re-engineered to be modular, and workflow driven. Also, over time, Helm's operations, user onboarding, and Self-Service capabilities were hardened. Profiles were formally introduced to aid with domain-agnostic features. The platform was designed to scale, in terms of data variety and volume. With additional success, multi-tenancy had gained spotlight. Helm instances were spun up to support Comcast syndication partners and other tenants.

Current Challenges and Future Work: Spinning up new tenant and partner Helm instances demonstrates versatility. However, standing up a new Helm instance can take several hours. Our ongoing initiatives aim to reduce the time it takes to standup new instances. An end-goal is to host Helm in a SaaS model using a services-first approach. Additionally, certain aspects of Helm are not managed by Self-Service. For example, direct data access from Helm infrastructure (AWS Kinesis streams and S3 buckets) requires a code deployment. Similarly, observability dashboards are not externalizable, and teardown infrastructure orchestrations are yet to be integrated into Self-Service. With regards to data quality, statistical data quality checks over longer time ranges (days or months) are not managed in Helm. This is a future capability that can help identify data drift for analytics and machine learning use cases. We are also in the process of evaluating hot/warm storage techniques to optimize infrastructure costs.

5. Conclusion

In this paper, we introduced "Helm", a CDP that enables employees to deliver best customer experiences by giving them a faster visual timeline of what's going on, when our customers are experiencing service issues. We showed how CDPs like Helm provide value by consolidating customer experiences from siloed domains or organization verticals.

Data management and governance are the biggest challenges to any customer data platform. The paper highlights the importance of a Self-Service-oriented management layer to govern all aspects of data. As a direct result, the Helm CDP platform is modular and integrated with constituent applications, to enable data re-use. We demonstrated how these applications, including Analytics, 360º Visualization, and Journeys, work together to understand customer experiences and address concerns. We described the design principles that influenced Helm and described its architecture. Finally, we shared the architectural evolution of Helm and future work.

6. Acknowlegements

Neither Rome nor Helm was built in a day. Helm, as it stands today, has evolved several times since its inception as a "Lab Week" project at Comcast in 2012. It has come to fruition from the incredible work of a driven team of less than hundred members. I would like to thank and acknowledge all the engineers, managers, architects, and leaders who have made Helm successful over the years. I would like to particularly thank Leslie Ellis, Dave Torok, and Edward McLaughlin for their valuable inputs on the paper.

Disclaimer: All errors, viewpoints, and opinions in the paper are from the author. They do not necessarily represent the position of the author's employer or affiliation.

Abbreviations

AWS	Amazon Web Services
API	application programming interface
CDP	customer data platform
ETL	extract, transform, load
GUID	globally unique identifier
HTML	hypertext markup language
HTTPS	hypertext transfer protocol secure
JSON	JavaScript object notation
MAC	media access control
OIDC	OpenID Connect
NPS	net promoter score
SaaS	software as a service
SLO	service level objective
SMS	short message service
SPI	service provider interface
SQL	structured query language
UI	user interface