www.comunityplatform.com

ComUnity Platform
Technical Description

comunity
smart, connected communities

ComUnity Platform Introduction

ComUnity revolutionises the way organisations succeed in a digital world. Our digitisation platform connects data, choreographs digital operations, and creates contextualised anywhere, anytime customer experiences to every device.

Our low-code, end-to-end platform automates multi-sided digital ecosystems, enabling organisations of all types to digitise operations more than 10X faster, better, and more cost-effectively.

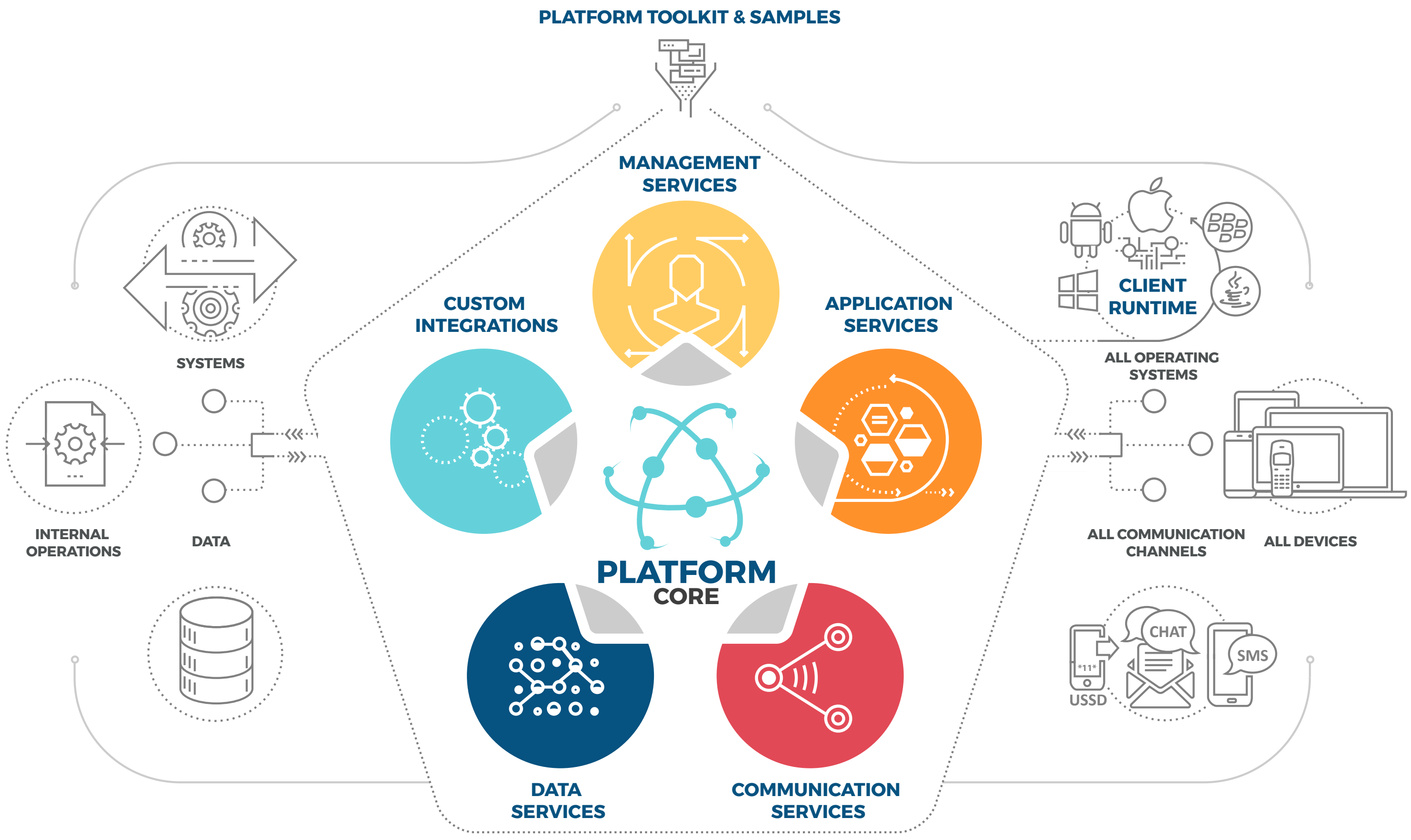


Figure 1: High-level map of the ComUnity Platform showing the various components and services.

The ComUnity Platform consists of a number of components and services which support delivery of the platform's omni-channel promise. These components work together to deliver a unique digital experience across multiple channels, devices and settings.

This document will describe these components, and their sub-components, in greater detail and illustrate how the core functionality is supported by the ComUnity Platform.

This core functionality is achieved through three key concepts:

Harness

The ComUnity Platform enables developers to plug into existing structured data, unstructured data and metadata sources and leverage these to build omni-channel applications off a single application build.

By using the Platform Developer Toolkit, a developer can follow a declarative approach to creating the server-side of the application and UX Designer to define the client UI. The ComUnity Platform will then generate the client builds and interfaces for all mobile, computing and web platforms including:

- Native Apps: Windows 10, iOS, Android, Feature phones (MIDP2)
- Web Apps: All mobile, PC and console browsers (HTML5 + XHTML MP 1)
- Social Media and other 3rd Party applications
- Internet communications systems: Instant Messaging, Chat & Email
- Telecommunications protocols such as SMS, MMS, USSD

Choreograph

The Platform Core, and associated Platform Services, comprise a next generation service delivery platform that deals with the packaging, distribution, delivery and management of omni-channel applications (see Figure 2). This is managed in line with the user profiles (who are they, what communities do they belong to, what services are they eligible for, etc.) and posture (channel, device, operating system, application platform, security clearance, etc.).

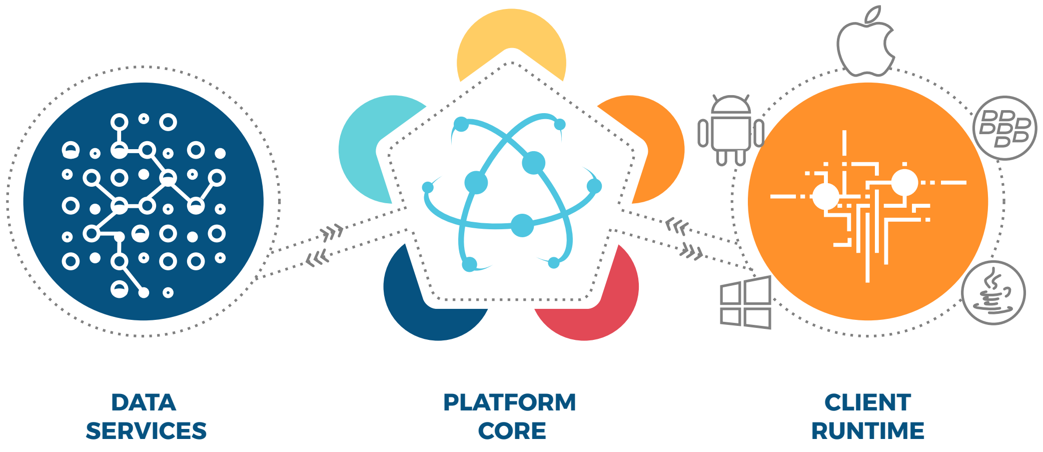


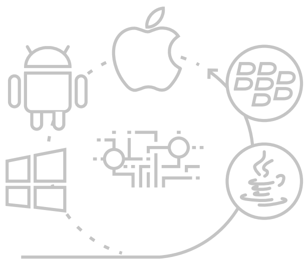
Figure 2: The ComUnity Platform Core, and primary Platform Services.

User interactions can be migrated, managed and tracked across access channels, providing a truly integrated and choreographed omni-channel user experience. The platform will always recommend and provide the best access experience dependent on the users' profile and posture, or preferences.

Proliferate

The ability of the ComUnity Platform to translate and manage application experiences to the broadest array of UX, and access scenarios, enables Platform tenants to rapidly scale the reach, relevance and impact of digital initiatives.

ComUnity's unique platform architecture rapidly increases the rate of innovation and responsiveness in servicing audiences over mass-market communications infrastructure. At the same time, there is great reduction in the cost of building and owning separate code bases for the multitude of access scenarios available to users in the digital age.



CLIENT RUNTIME

The client runtime is responsible for rendering the user interface of applications for a specific target device. These applications are built using the Platform Developer Toolkit and stored in the application repository. The client runtime abstracts the user interface so that the application only needs to instruct what information needs to be displayed or captured and what the navigation hierarchy is. The client runtime is responsible for how the interface is rendered and how the hierarchy is navigated. Instructing the 'what' instead of the 'how' gives the client runtime an enormous amount of freedom in optimising the implementation for the target device.

For instance, the client runtime may decide to change a scrolling form to a paging form because of memory constraints. What's important is that the correct information is captured in the most user-friendly way, not how the data is captured. The application declares what it needs and gets out of the way for the client runtime to execute.

The only coupling between the client runtime and the platform core is the start URI and the ComUnity application protocol. This application protocol is relatively simple and consists mostly of HTTP idioms and media-types. This de-coupling of the client runtime, the platform core and applications allows independent evolution of client runtime, server infrastructure and user experience technologies. It allows 3rd parties to implement their own competing client runtimes or use ComUnity client runtimes and implement the server infrastructure.

Installable client runtimes are delivered either via the Application Logistics Manager, or via the relevant application store (i.e. Apple App Store, Google Play, etc.). When using a browser to access application functionality on the ComUnity Platform, the platform detects alternative client runtime choices and presents the options to the user. The client runtime will either be served by the ComUnity Application Logistics Manager, or the user will be redirected to the vendor specific application store.

It is important to note that a key property of all clients is that there is little to no application specific knowledge built into the client. The user interfaces are almost entirely driven by models served by the platform.

PLATFORM CORE

The ComUnity Platform Core contains a set of shared components which serve as the “central nervous system” of the digital operating platform. While some of these components perform centralised tasks within the core, others are responsible for providing a common set of services across all platform service layers.

At a high level, the Platform Core responds to requests from the Client Runtime and hosted applications.

The Platform Core will make sure the interacting entity is who it says it is by calling the authentication component, that the entity has the appropriate permissions by calling the authorisation component and that it is possible to satisfy the request. The core will then call on the necessary services to assemble the response.

If the request is for data only, then the request will be routed through the components of the Platform Core to the Data Services. If the request is for metadata, then the request will be routed to the Application Content Manager.

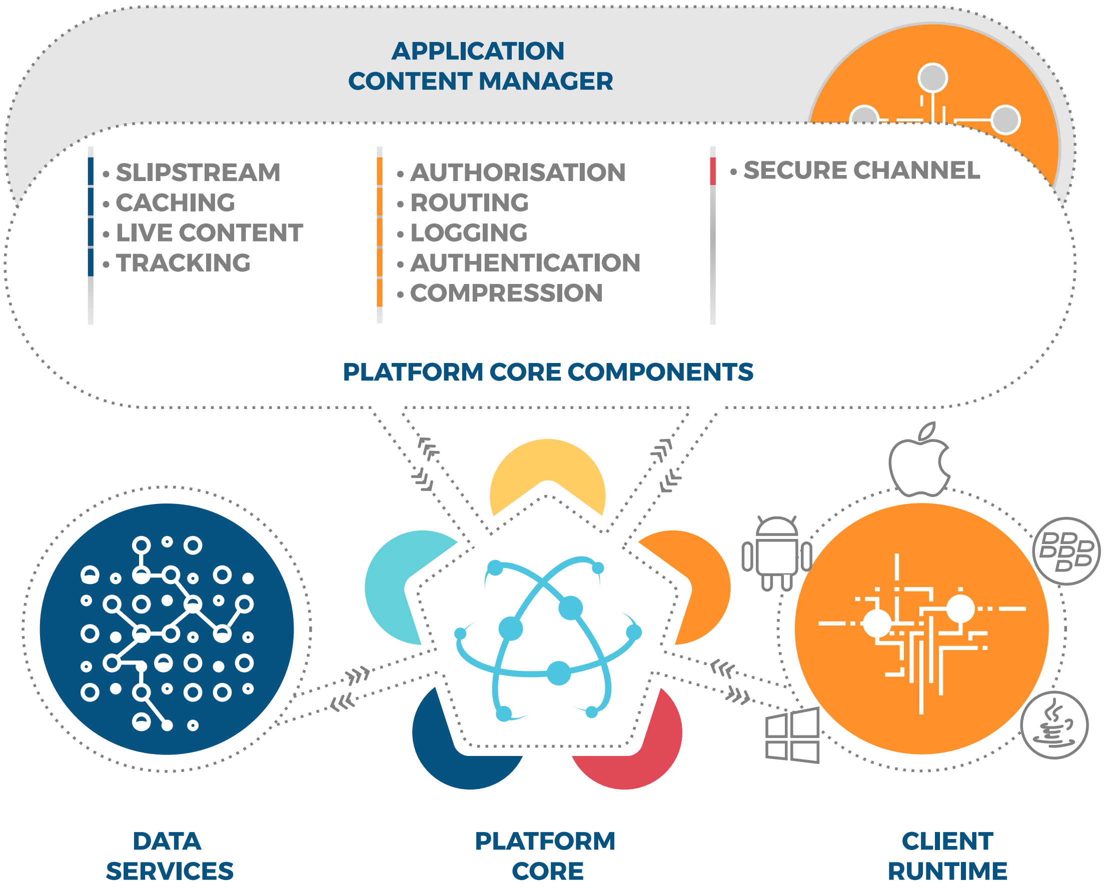


Figure 3: Details of the Platform Core and interaction between Client Runtime, Application Content Manager and Data Services

PLATFORM CORE

The various Platform Core components are discussed in more detail below:

Secure Channel

Secure channel is the first component of the Platform Core, and provides message privacy, tamper protection and server authentication. All devices that support secure channel communications will use this component.

Compression

The compression component detects user agent support for compression and compresses the response. This applies to both media and data although many content types like JPEG are delivered as-is since the format is inherently incompressible.

Authentication

Whenever the incoming message asserts an identity, the identity is authenticated against the authentication service. The authentication service allows authenticating with an e-mail address, telephone number or a ComUnity account name.

Logging

The logging component records the request's metadata in file storage. When the response comes back its metadata is also logged. Sensitive data like authorisation data is removed before persisting. When an error occurs, the request and response bodies are also stored. All log operations are asynchronous to minimise the effect on responsiveness.

Routing

The routing component examines the incoming request and passes it to the appropriate handler. Media and data requests are first passed on to authorisation while tracking and metadata are passed to their respective handlers.

PLATFORM CORE (CONTINUED)

Authorisation
The responsibility of this component is to authorise the request and then return the user's profile including security roles. The authentication and authorisation layers ensure that any operation on any resource happens with permission. When the incoming request does not satisfy these requirements, an error is returned to the client. The client applications handle such errors by presenting the user with a choice to sign in to continue.

This reactive authorisation also caters for the scenario where the client application needs user profile information to render UI elements or the client application detects that the user's security roles are not sufficient for the targeted resource. In these cases, the login service can authorise new credentials and return a user profile with possibly elevated security roles.

Tracking
The tracking component records metadata about the request and then passes control to the authorisation component after setting the request URL to the URL of a resource configured in the tracking record identified by the original request URL.

Live Connect
The live connection component is responsible for delivering changes to a resource as quickly as possible to clients that are interested in them. Each request, and its response, is assigned a unique id to enable tracking of the request as it moves through the request pipeline.

Caching
Based on cache control metadata the caching component may decide to invalidate current cache entries or deliver a response from cache storage. When a request is passed on, the resulting response is evaluated for cache control instructions to determine whether the response may be recorded in the cache. The caching component supports expiration and validation.

Slipstream
The slipstream component makes sure that there is only one idempotent request in-flight for the same resource on the same processor. When a request comes in while another is in progress for the same resource, the requests are added to the queue of requests waiting on the response. When the response comes back, the same response is delivered to all the requests at once.

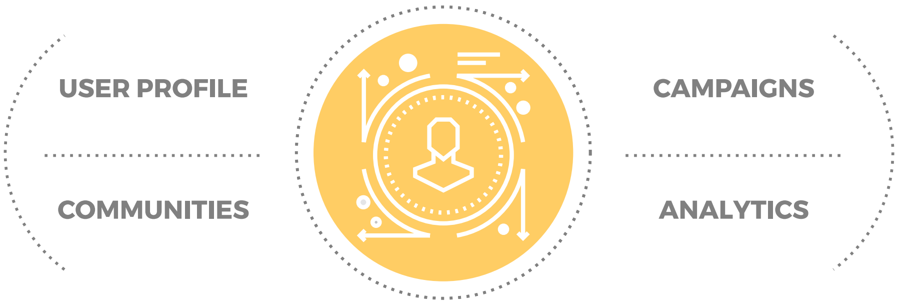
The slipstream component can greatly increase scalability during peak times.

MANAGEMENT SERVICES

These components are discussed in more detail below.

Authorisation

The management, control, and coordination of all Digital Platform tasks are facilitated within the Management Services layer. This includes back-office tasks such as managing users, building out digital campaigns and gaining insights into the digital ecosystem via analytics dashboards.



MANAGMENT SERVICES COMPONENTS

User Profile

The configuration of the User Profile, and management of the various authentication and authorisation parameters, lies at the heart of the ComUnity Platform. The platform consists of a core user profile which may be extended in individual projects to enhance user-related functionality.
In addition to controlling server-side permissions, the user profile component is used to manage client-side UI rendering and navigation capabilities.

Communities

Many of the application and communication features of the platform are managed via the Communities component. A user may be configured as a member of one or more communities, which will in turn control access to targeted information, campaigns and application functionality.
Communities may include geographic segmentation (e.g. city or suburb) or any other measures such as group-related affiliations, organisational memberships and hierarchies.

Campaigns

Various user-based, cross-channel digital message flows may be controlled via the Campaign Manager. This capability allows targeting of the appropriate audience through communication channels, applications and social media.

The Campaign Manager includes content authoring, customer journey mapping and flexible scheduling.

Analytics

The ComUnity Analytics captures application usage patterns, user preferences, community-based platform utilisation and communication strategies, allowing for the rapid derivation of insights based on underlying digital data.

APPLICATION SERVICES

Application Services provide a set of capabilities which support application-related activities, and comprise:

- Application Logistics Manager: a next generation delivery platform that deals with the packaging, distribution, delivery and management of omni-channel applications.
- Application Content Manager: a set of services which manage the real-time metadata and content-related capabilities of the application platform.

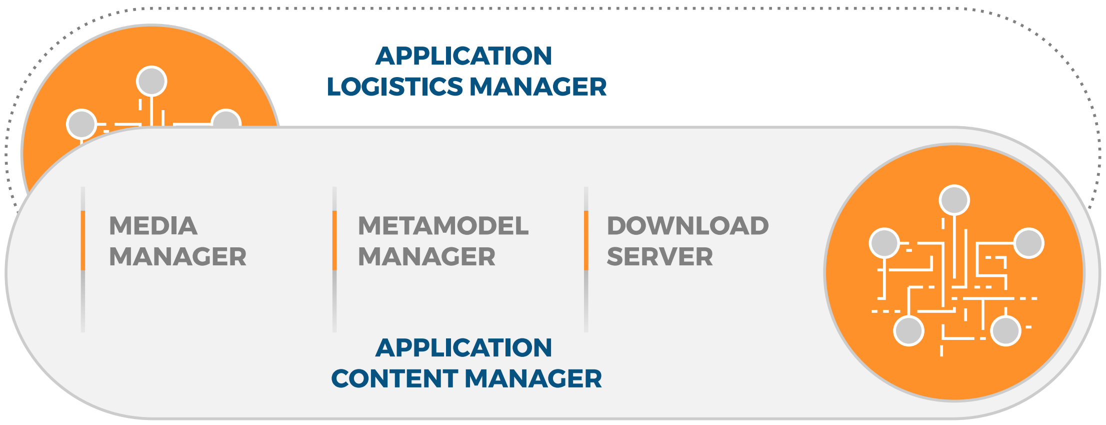


Figure 5: Detailed components of the ComUnity Application Services

APPLICATION SERVICES

Application Logistics Manager

The ComUnity Platform includes a number of capabilities which support the distribution, management and operation of complex omni-channel application ecosystems. These include a sophisticated distribution system, processes to automate delivery and updates of native applications and capabilities to control delivery of application metadata packages.

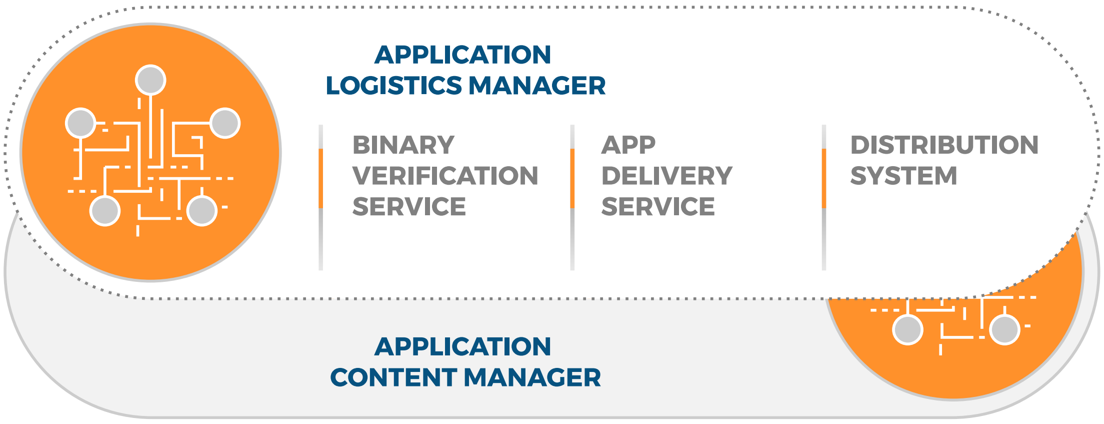

APPLICATION LOGISTICS MANAGER
BINARY VERIFICATION SERVICE
APP DELIVERY SERVICE
DISTRIBUTION SYSTEM
APPLICATION CONTENT MANAGER

Figure 6: Components of the Application Logistics Manager

Binary Verification Service
Client applications automatically check for new binary executables when they come to the foreground. The client interacts with this service when performing the binary executable version check and this may result in a message prompting the user to download a new version.

Application Delivery Service
When applications are built with the Platform Developer Toolkit they are stored in the application repository. Delivering an application includes the application's form definitions, navigation hierarchy, styling information and graphical assets.

The Application Delivery Server detects the incoming device, determines the options available for the user to experience the application functionality and renders the appropriate UI for the user to make the choice. This allows the developer to design once, yet ensures applications will be rendered accurately, quickly and with minimal errors across a number of target devices and platforms.

Distribution System
The system can push application links to users by sending encoded personalised links via SMS or email, and can also pull users by publishing encoded links on the Web, Facebook, etc. The links direct the users to the ComUnity Platform where the Distribution System automatically detects the most appropriate client runtime for the user. If an appropriate runtime cannot be found, the user is redirected to the appropriate web application.

Personalised links are decoded, and the download log updated with the user's identity and the distribution campaign.

When a user reacts to a push link, the system knows who the incoming user is, what application is requested, and on which campaign the user is reacting. When a user reacts to a pull link, the system knows which published link was clicked on and which application is targeted, but the user remains unidentified until login time.

APPLICATION SERVICES

Application Content Manager

The Application Content Manager responds to requests originating from the Routing component of the Platform Core. If the request is for media, the core will hand it off to the Media Manager. If the request is for application-related metadata, then the Metamodel Manager will deliver the combined metamodel to the client.

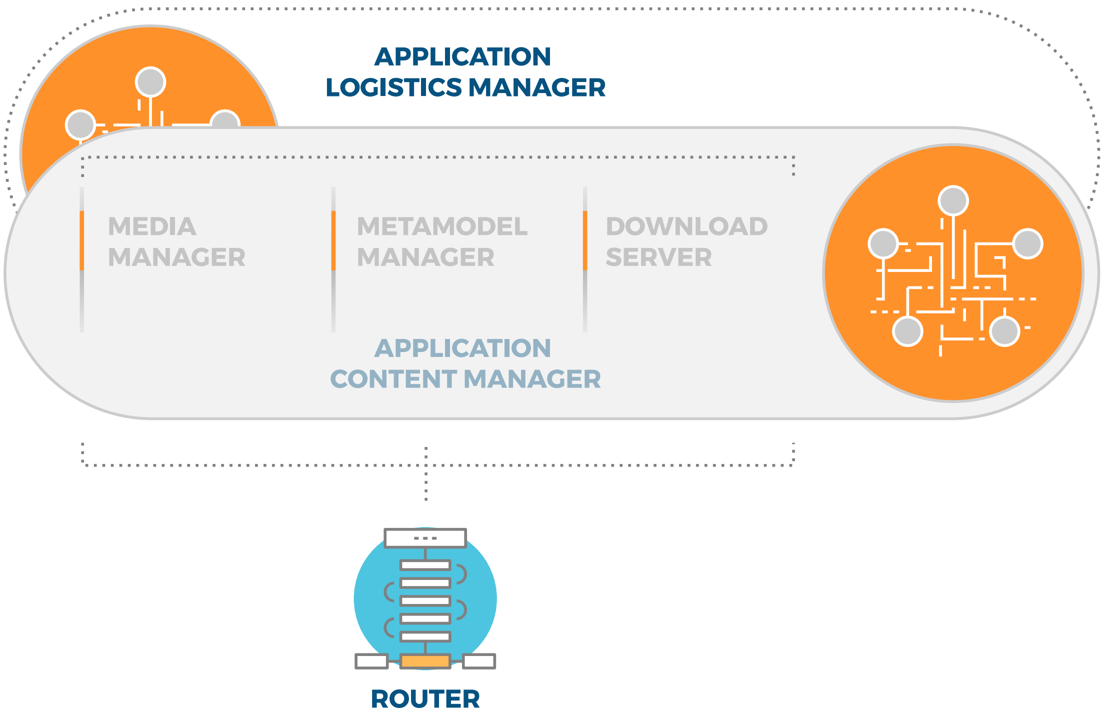


Figure 7: Application content routing mechanism in ComUnity Platform

Media Manager

The media manager consists of a set of capabilities dedicated to processing, storage, and performance improvements associated with delivery of digital media content (documents, photos and other files) to requesting clients.

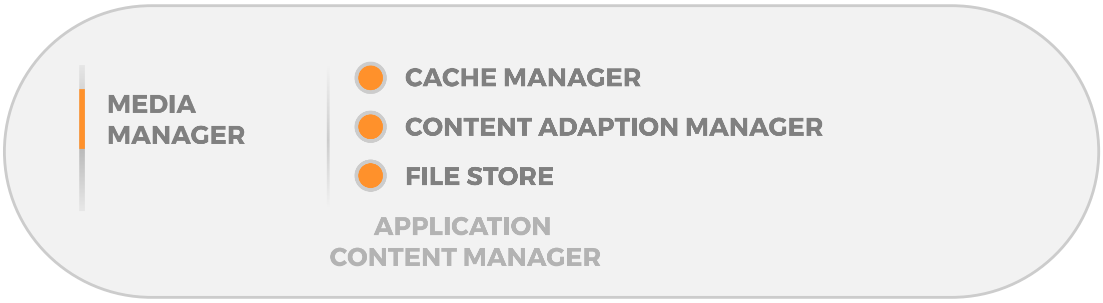


Figure 8: Components of the Media Manager

Cache Manager

The media server Cache Manager supports cache expiration and validation of media assets to increase performance and scalability of media-related request operations.

Content Adaptation Manager

This component intelligently serves the appropriate resources requested from the media service. If the resource is a JPEG or PNG image, the media server also supports real-time resizing of the image, or creating a thumbnail of the image.

File Store

The File Store component is responsible for storage and retrieval of documents, images and any other files required as part of application content management.

Metamodel Manager

When a client requests application-related metadata at run-time, the application is asked for its data model via the Data Model Service, while the Metadata Service is asked to provide the application metadata. These are then packaged together as the metamodel and delivered to the client.

At design time, a developer would use the Platform Developer Toolkit to link the application metamodel and data models. One of the ways these resources are linked together is by the developer using the UX designer when the user interface of an application is defined.

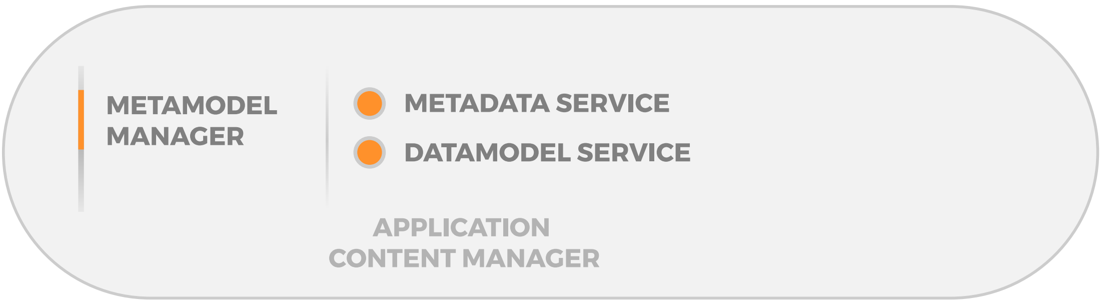


Figure 9: Components of the Metamodel Manager

Metadata Service

Metadata refers to assertions made about information. In the ComUnity Platform metadata is used in the models that describe the information which applications care about, and the user interfaces used to interact with that information. The Metadata Service hosts the metadata that enables client applications to weave the data model into user experiences.

This metadata is created at design time by a developer using the Platform Developer Toolkit.

Client applications automatically check for new metadata when they come to the foreground.

Data Model Service

The Data Model Service helps to manage and describe the data model including the entities, properties, associations and the navigation between the entities.

An important benefit of the ComUnity Metamodel driven architecture is that UX changes can be done in real-time via updating the underlying metadata and/or data model. Such changes will simply require that the native application automatically refresh the metamodel in order to implement changes, and will not require an application package file update from the various App Stores (e.g. .apk for Android and .ipa for Apple. This capability dramatically accelerates deployment of application changes.

COMMUNICATION SERVICES

The Communication Services are responsible for controlling all communication channels including telecommunications protocols, internet-based communication, social media and application messaging. Such capabilities allow for the choreography and management of complex cross-channel message flows which characterise the modern digital customer journey.

The Communication Delivery Manager abstracts the underlying communication channels from the delivery processes and allows for the implementation of generic notification patterns based on:

- user profile: who are they, what communities do they belong to, what communication preferences have been configured etc.
- posture: available channels, device, operating system, security clearance, etc.
- policies: severity, failover, re-tries etc.

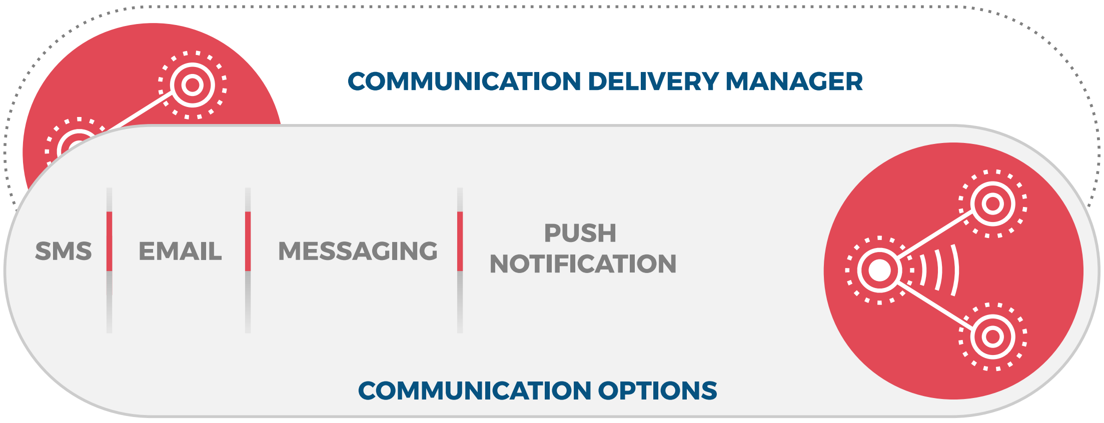


Figure 10: Communications Services Components

The Communication Delivery Manager can co-ordinate and correlate messages which may be outbound only, inbound only, or 2-way. In addition, various communication patterns are supported such as:

- Fan Out: a single message is created and pushed out to all subscribers on single channel (e.g. SMS)
- Preferred Device: messages are delivered based on user profile, available channels and/or policies.
- Auto Fall-back: if message delivery fails on one channel, a retry will be attempted on another channel.

A developer would use the Platform Developer Toolkit to implement design-time services which leverage the Communication Delivery Manager using common communication services. For example, a particular design-time process may call a “Notify” method while the run-time process would leverage the Communication Delivery Manager to choose the appropriate delivery channel/s and communication pattern based on configuration policies.

DATA SERVICES

The Data Services provide the ability to expose data as a service using the Open Data Protocol (OData) via the semantics of representational state transfer (REST).

- The first responsibility of the data service is to provide an interface to the storage engine hosting the application's data. This application database allows building a full end-to-end user experience without any external dependencies, running 100% on the ComUnity Platform.
- The second responsibility is to provide a description of its data model including the entities, their properties and the navigation between the entities. This description is what the Metamodel Manager uses to bootstrap the application's UI.

For data exposed by the application Data Service, the data is automatically linked to other resources by the relationships in the data model and exposed via the Data Model Service.

If so designed, an application hosted on the ComUnity Platform can access all the data-related functionality it needs inside Data Services. At other times, an application may need to integrate with external systems. In the case where the data services are just a façade in front of another external service layer, the service calls need to be intercepted and passed on to the external layer – the task of the Custom Integration Services.

In the current version of the ComUnity Platform, these Data Services may be created automatically using the ComUnity Platform Developer Toolkit. The resultant Data Services project may then be manually customised using Microsoft Visual Studio using Entity Framework for the Object-Relational Mapping (ORM). The default back-end data store is Microsoft SQL Server.

CUSTOM INTEGRATIONS

Integrating with an external system typically involves creating an adapter that makes the external system look like a ComUnity application service.
When integrating with external systems it helps to summarise the relevant properties of the ComUnity Platform.

- The platform has a resource oriented view of the world with identifiable resources linked together where a standard and limited set of operations is valid on any resource.
- A stateless back-end
- Often, external systems don't have the properties summarised above and so require some elaboration in order to understand the process of integrating them efficiently into the ComUnity ecosystem.

www.comunityplatform.com

United Kingdom

8 High Street,
Brentwood,
Essex,
CM14 4AB

Tel: +44 7577 001145

South Africa

Spaces, 377 Rivonia Boulevard,
Rivonia 2128,
Gauteng,
South Africa

Tel: +27 11 593 2428

comunity
smart, connected communities

V8.6