tomtom

NavKit2

The next generation of cloud-native navigation

Whitepaper

Introduction

NavKit2 is TomTom's next-generation, cloud-native navigation engine. It is the core technology that powers our complete navigation offering, including maps, connected services and user interface (UI) — delivering an in-vehicle experience that no smartphone app can match.

Navkit2 is online-first, enabling responsive search, routing and guidance. When needed, it switches seamlessly between online and onboard without any disruption, creating a safer, more comfortable driving experience.

This whitepaper explains how NavKit2's architecture enables developers to build navigation solutions with a wide variety of hardware and connectivity configurations. It is a conceptual description rather than a technical specification, so the information in it is high-level and its examples are somewhat simplified.

Although NavKit2 can also be used to develop consumer software, this paper focuses on the automotive sector and is intended for readers in that industry.

Navigation in context

The problem

There is no question that the “lap map” is dead. Today’s drivers instinctively reach for electronic navigation tools, either brought in or built into a vehicle, to get where they want to go. These tools have changed considerably over time as the market has grown and become more sophisticated. TomTom introduced its first Portable Navigation Devices (PNDs) in 2004, and its first in-vehicle navigation system in 2009. Since those early days, new routing options, alwaysfresh traffic information, better map search, more detailed guidance information, and advanced vehicle location data have all made driving easier and safer. However, one thing has not changed. We still think of navigation software as being “in one place”: No longer on a paper map, but in a car’s head unit, on a phone, or in the cloud. Each location has its advantages. Online processing is fast, and can take advantage of always-fresh maps, data and functionality. Onboard navigation keeps drivers on the move even in areas with no connectivity. Mobile phone software is easy to keep up to date. Head units are the natural home of driver interfaces. But each of these options also has disadvantages, which translate into constraints on driver experience. For a long time, navigation systems have been restricted by where their components run. That no longer needs to be the case.

The solution

NavKit2 breaks this last constraint. Developers can use NavKit2 to create navigation systems that take advantage of the freshness and speed of online processing, and have the local resources to keep drivers moving when online data isn’t available. Automotive manufacturers can build applications that are as distributed as the rest of the software in a modern vehicle: Collecting data in one place and processing it in another, displaying different maps and navigation data in different places at the same time.

Designed for flexibility

NavKit2's design is built around a loose coupling of components. Wherever possible, modules run separately from one another, with clean, well-defined and flexible interfaces between them. This approach provides developers with numerous options for designing and deploying navigation systems.

Modularity

The functional areas within NavKit2 are designed for modularity. Each navigation function (such as map display or route planning) is as self-contained as possible. Even when two functions share resources (for instance, an onboard map), they do not depend on each other.

This means that customers can pick exactly the functionality that supports their users' needs. It also allows them to easily combine TomTom's services with those from other sources, so that a navigation application could be built using TomTom's guidance technology and the customer's own map display functionality.

Clients and services

Within its modules, NavKit2's architecture uses a client/service model:

- A collection of client libraries controls how user-facing functions interact with the services that power them. Developers can use these libraries to build user interfaces that suit their customers' needs.
- The navigation functionality itself is provided by a set of specialized onboard services. These services perform functions like interfacing with information on the device, performing onboard calculations, and managing interactions with online services.

As an example of how this works, a simplified view of the components involved in map display and

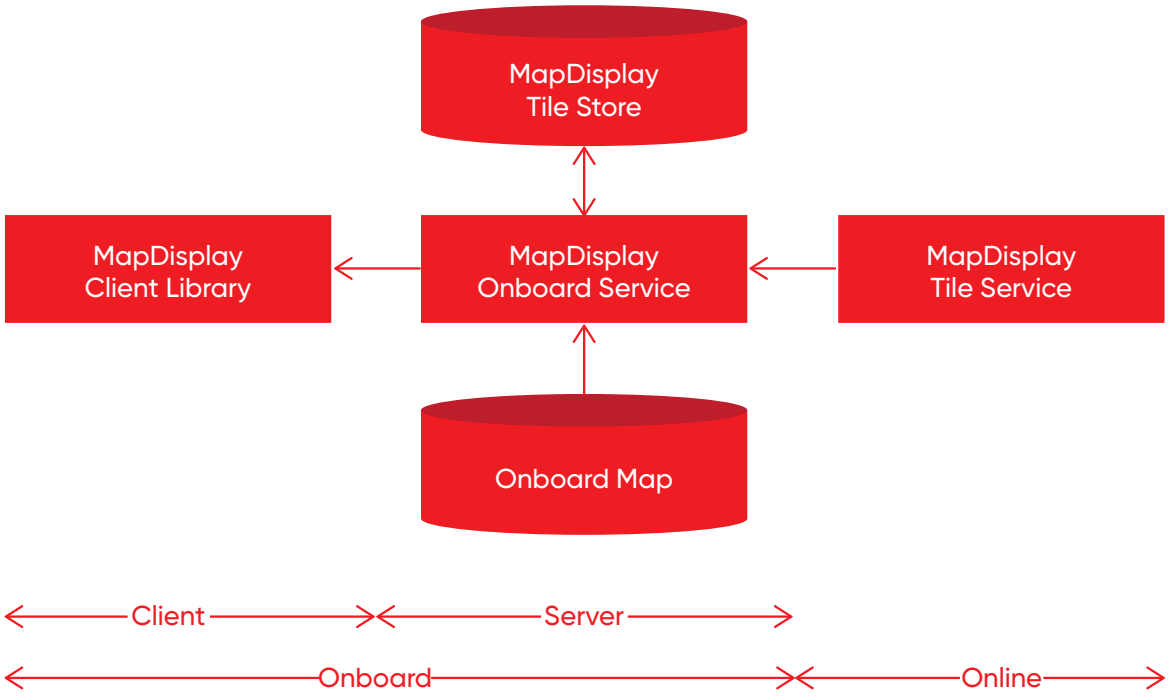
Map display in action

As an example of how this works, a simplified view of the components involved in map display and interaction is shown below.

The **MapDisplay Client Library** gives developers access to user interface elements such as camera controls, touch and gesture interactions, and drawing custom elements on the map.

The client library interfaces with the **MapDisplay Onboard Service**, which provides access to map data. This data can come from the online **MapDisplay Tile Service** or from the **Onboard Map**. In both cases, the onboard service manages the connections with the data sources and returns whatever the client has requested. When the data source is online, the service also caches downloaded data in a local **MapDisplay Tile Store** to avoid repeat calls.

Note that more than one client can access the same service. Each client uses the service to obtain whatever data it needs to create its own interface. For example, a client running in the center stack can use the MapDisplay Onboard Service to support route planning and guidance. At the same time, a different client can use the same service to show progress along the current route in a rear seat display, while a third client in the cluster provides details of an upcoming maneuver.



For a more complete picture of NavKit2's component architecture, see Appendix A.

Flexible communication

Modularity and loose coupling make navigation system designs more flexible, but the different elements still have to be able to communicate smoothly and securely with each other. Communication can impose its own limitations on systems design.

NavKit2 removes those constraints with its own remote procedure call (RPC) mechanism for communication between its different components. Based on Google's protocol buffers, RPC allows modules to communicate with each other over a variety of different transport layers without impacting their functionality. The transport layer is chosen based on where the components sit in relation to one another, as seen in the table below.

Using a transport layer-agnostic protocol means that components can run at any distance from each other within the vehicle, from sharing a process to being deployed on separate pieces of hardware.

This kind of distributed deployment allows developers to use NavKit2 out of the box to create application architectures such as these:

- A single app made up of clients and services sharing the same processes.
- An app made up of client libraries that call independent services running on the same hardware.
- An app running on one piece of hardware calling services that run on other hardware within the vehicle.
- Apps that combine the above features.

Component proximity Features	Transport layer(s)
Components run in the same process or the same app	In-memory
Components run in different applications on the same hardware	TCP/IP, Unix Domain Sockets
Components run on separate hardware	TCP/IP over a network connection

Online and onboard Functionality

Online first

NavKit2 is online first, meaning that it is designed to NavKit2 is online first, meaning that it is designed to create applications that use the richness and speed of online services by default. Developers can use it to build high-quality systems that rely completely on TomTom's range of online navigation services.

Like other online apps, software built with NavKit2 caches downloaded data to reduce repeat calls, and thus bandwidth usage. NavKit2 caches navigation and map data; developers can configure how much data is kept and for how long. This cached data can also be used to bridge unexpected connectivity gaps.

Onboard fallback

Online first does not necessarily mean online only. If the application's cached data is expired, or is not relevant to the vehicles's current position, the application has to fall back to whatever resources it has onboard. In practical terms, that means an onboard map.

Onboard maps
Before online map services became available and better data plans made them affordable, onboard maps were the default. They have since fallen out of favor, because of two main problems:

Footprint
It is hard to fit enough map data into an acceptable device footprint to deliver a good default customer experience.

Updates
The more map data there is on a device, the larger the updates have to be to keep that data current. Because traditional onboard maps cover more territory than most drivers visit, much of the downloaded update data is never used.

TomTom leverages the latest mapping technology to reduce these problems, so that NavKit2 can provide the option of an onboard map for fallback navigation. Like its online maps, TomTom's onboard map for NavKit2 follows the Navigation Data Standard (NDS). NDS allows TomTom to control how much information the map contains about each geographic area, which in turn affects how much storage is needed. With NDS, onboard maps can also be divided into many small regions instead of a few large ones. This granularity, combined with smart policies about what regions are stored, further reduces the map footprint. Rather than try to keep the whole onboard map continuously up to date like an online map, NavKit2 uses smart streaming to automatically update only the most relevant parts of the onboard map, in the most efficient way.[1]

1 See TomTom's Map Management White Paper for more details

Navigating while disconnected

Below are the experiences of two hypothetical users as they plan and navigate routes in different circumstances. Together, their experiences show how software built with NavKit2 can operate online, in comparison to how it can work offline with and without an onboard map.

These steps are not a complete description, and they have been simplified to make the examples clearer.

- **Amina** has an application that provides both online functionality and an onboard map. The onboard map means that her application takes up more space on her device, and the map data has to be updated from time to time.
- **Ben** has an application that supports the same online functionality as Amina's. To save space and data, it does not have an onboard map.

1. Choosing a destination

Below are the experiences of two hypothetical users as they plan and navigate routes in different circumstances. Together, their experiences show how software built with NavKit2 can operate online, in comparison to how it can work offline with and without an onboard map.

Amina and Ben can control many aspects of the routes they want to take, from the type of route (faster, shorter, more thrilling) to what kinds of roads to avoid. But the most basic choice is the destination: where are they each going?

- If they are connected to online services, they have a variety of ways of choosing a destination, including but not limited to:
- Using online search, including points of interest (POIs) and brands.
- Tapping on the map to select a location point.
- Selecting a previous destination from their travel history.

If **Amina** loses connectivity, she can use her onboard map and the personal data on her device to do some of the same things that she could online:

- Tapping on the map to select a location point.
- Selecting a previous destination from her travel history.

If Ben loses connectivity, he has no onboard map to fall back to. He can select a destination from his travel history, but unless he regains connectivity, he cannot plan a route to it.

2. Planning the route

When Amina and Ben plan routes with the online services, the algorithm combines their input and preferences with map data and time-sensitive information such as temporary speed limits and road closures. Taking all of this into account, it finds the best route for each of them. The algorithm considers current and historic traffic information, because the fastest road on Sunday nights may be jammed on Friday afternoons.

Amina can still plan a route if she loses connectivity. She has access to the same market-leading algorithm as she did online. The results not be perfect, because less data is available. Her onboard map may not be completely fresh, depending on her app's update settings. And the route will not include temporary conditions such as road closures, temporary speed limits, and current traffic incidents. However, because her onboard map includes TomTom's IQRoutes™ historic traffic information, her route planning and arrival estimates will still incorporate realistic road speeds and avoid predictable traffic.

Without connectivity or an onboard map, **Ben** cannot plan a route.

The next generation of cloud-active navigation

3. Navigating along the route

As Amina and Ben follow their planned routes while connected, they see a map of their current area, overlaid with information about upcoming maneuvers. Turn-by-turn instructions are provided at the appropriate times, along with other local information such as traffic jams on the road ahead. If they briefly lose connectivity, what they see does not change as long as their cached information is valid and relevant to their location.

If she continues to be disconnected from online services, eventually **Amina's** application will use the onboard map instead of the online one. She will still see her progress along her route, and will still receive guidance information such as turn by turn instructions. The map around her will be visible, but there will be no traffic information on it.

If he stays offline after he leaves the area that his application has cached data for, **Ben** will only see the route shape and current position on a blank background. He will not receive any guidance.

4. Replanning en route

Because traffic and road conditions change constantly, NavKit2 periodically replans Amina's and Ben's trip online to see if a different route is better for them. If their vehicles leave the planned route, a new route is calculated that will bring them to their final destinations from their new locations.

If **Amina** loses connectivity, her application will only replan her route if she leaves the current one.

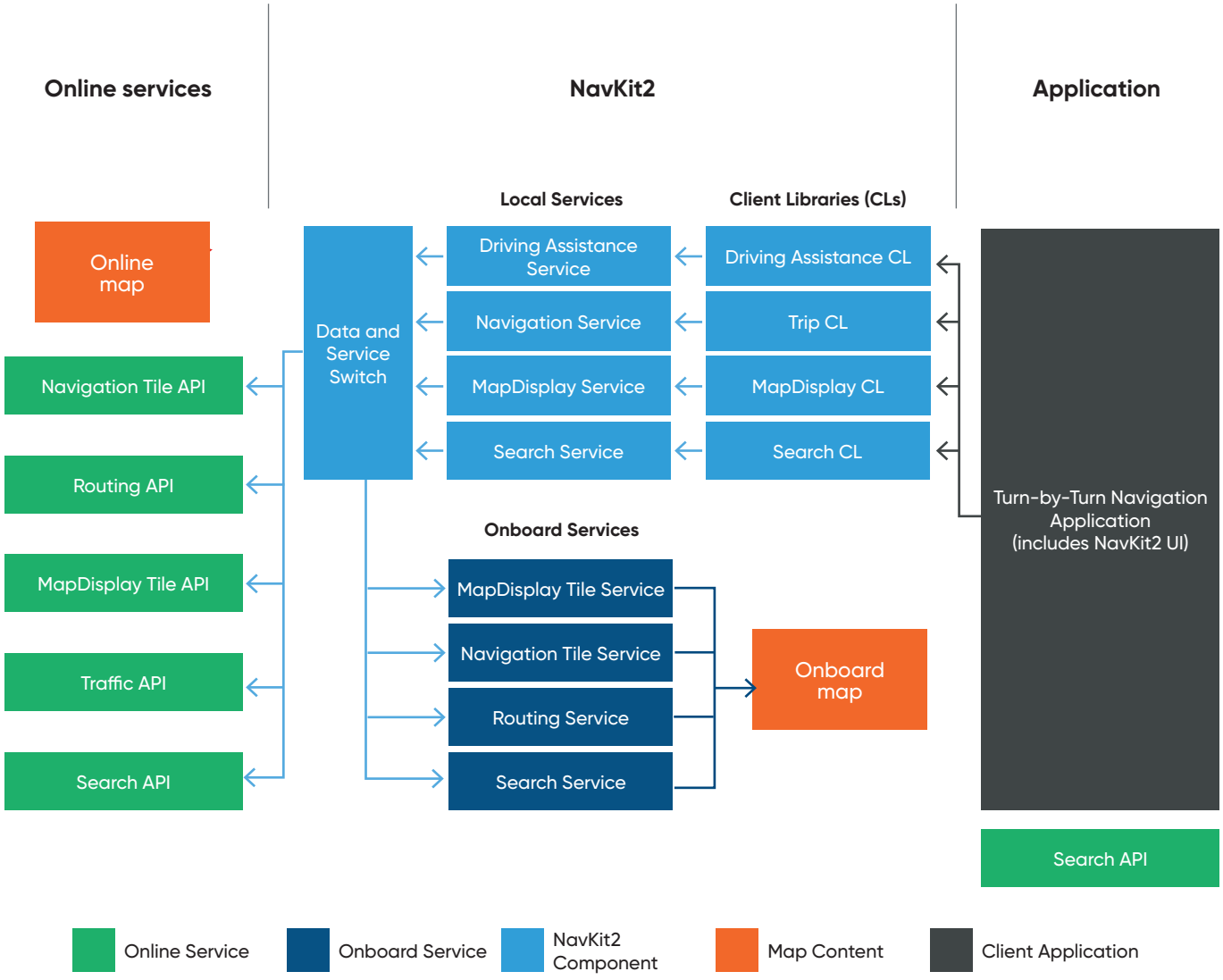
Because her onboard map information will not change as she drives, there is no point in replanning while she is underway.

Ben's application will not replan his route, even if he leaves the current one. Without an onboard map, there is no data to plan with.

Navigation for a new generation

NavKit2 allows automotive system designersand developers to create a new kind of navigation software whose design is not limited by where its components are deployed in the vehicle. It helps them to build systems that combine the best of the online and onboard worlds. With NavKit2, automotive customers can stop thinking about where the navigation system

Appendix A: NavKit2 architecture



tomtom

For more information visit TomTom.com