

Mobile COBOL Developer Guide

Welcome to Mobile COBOL

Picture this: It's a Tuesday morning. Coffee in hand, you settle into your chair, fire up your trusty COBOL editor — the same one you've used for years — and get ready to tackle today's project.

Only this time, something's different. You're not building a batch report or updating a decades-old payroll system. You're about to build... a mobile app.

And you're going to do it without learning Swift, Kotlin, or any new language. No wrestling with Xcode. No Android Studio labyrinths. Just COBOL. The same COBOL you've mastered and known and loved all these years.

That's Mobile COBOL.

We created it for people like you — real developers with real code, who don't want to mess with the maze of mobile SDKs, JavaScript frameworks, or whatever the latest fad is today. Mobile COBOL is different. It lets you write mobile apps with plain old COBOL — the language you already know.

The magic? A generic mobile client that connects to your COBOL process and turns your COBOL calls into a living, breathing mobile interface.

Behind the scenes, the generic client is built using Microsoft's .NET MAUI framework — the same cross-platform technology used in modern mobile apps. This means you have access to a wide range of control types, properties, and events from COBOL, so you get a full-featured UI without learning .Net MAUI itself.

What You Need

Let's keep this simple. Here's what you need to build your first Mobile COBOL app:

- A Windows machine (for now, Linux is coming in the next release) — nothing exotic.
- Your favorite COBOL compiler — seriously, any one you like.
- The MOBILECOBOL.DLL file — this little guy does all the talking.
- The MobileCoClient app on your mobile device. A generic mobile client app — already built, no changes needed. Just download it from the app store (iOS, Android, even Windows!)
- One tiny environment variable: MOBILECOBOLPORT — this tells your app where to listen.

No IDEs, no weird plugins, no mobile SDK installations. Just you, COBOL, and a few files.

This low code, low learning curve approach will let you build your first mobile app in minutes!

Your First Mobile COBOL App

You start the same way you always do. Begin with the traditional sections and just add the mobilecobol.ws copy file to the WORKING-STORAGE SECTION. In the PROCEDURE DIVISION do your normal initialization (open files etc.) but then you add something new:

```
CALL MC USING CREATEPAGE "Name=Page1|".
```

```
CALL MC USING ADDCHILD "Type=StackLayout;Padding=20;Spacing=15|".
```

```
CALL MC USING ADDCHILD "Type=Label;Text=Hello|".
```

```
CALL MC USING SHOWPAGE.
```

Those four lines create a mobile UI. A page, a layout, and a friendly label that says Hello. And just like that, your app is alive — on an actual phone. Of course there is a LOT more you can do and we will cover it all.

The syntax is simple, readable and consistent (just like COBOL) and we will cover the details shortly. Curious about the | at the end? That's just a marker. Most versions of COBOL don't pass the length of the CALL arguments, so we use the | to say, 'That's the end!.' Simple and effective.

How It Works

Under the hood, it's all very cozy.

The MOBILECOBOL.DLL handles the communication. When your COBOL app makes an API call, the DLL opens a connection (if it's not already open), waits for the mobile client, and then sends your instructions to it. The mobile client reads your name/value pairs and builds the UI.

The structure is easy: you create a page, add objects to the page, then show the page. You control everything with plain strings like Type=Label or Text=Good Morning.

The client handles the rest.

Running the App

You're almost there. Let's put it all together:

1. Set the MOBILECOBOLPORT environment variable to something like 7051.
2. Compile and run your COBOL program.

3. The MOBILECOBOL.DLL starts listening.
4. Launch the generic mobile client.
5. The client connects — and boom — your UI appears.

You can tap the button, enter text, switch pages. Each event gets sent back to your COBOL program as a little message. Event-driven COBOL? You bet. That's just one of the concepts we will explore in the next section.

Core Concepts

Before we dive into the deep end of API calls, let's take a moment to talk about the *big ideas* that make Mobile COBOL tick. Think of this as your "orientation day" — except instead of bad coffee and awkward icebreakers, you get clear concepts and a fast track to success.

1. Pages & Navigation

In Mobile COBOL, every screen your user sees is called a **Page**.

This first release uses **stack navigation**. Imagine a stack of dinner plates: you put a new plate (page) on top to show it, and you take it off the stack to go back.

Why stack navigation? Because it pairs beautifully with the way COBOL programs naturally flow — top to bottom, step by step, nice and orderly. Think of it just like CALLing another COBOL program and then exiting it.

2. Layout Containers

Every Page starts with a **layout container** — think of it as the frame you hang your picture in. Common choices are StackLayout (vertical or horizontal stacking) and Grid (rows and columns).

Behind the scenes, Mobile COBOL keeps a *container stack*. When you add something, it goes into the container on top of that stack. When you're done loading a container, you can call ENDCONTAINER to pop it off and work with the previous one.

3. Controls & Properties

Controls are the building blocks of your UI — labels, buttons, images, etc. Each has **properties** you can set, like Text, FontSize, or Padding.

4. Events

An **event** happens when the user interacts with your app — tapping a button, scrolling a list, typing into a box.

Mobile COBOL sends you two pieces of event info:

- MC-EVENT-CONTROL → The Name of the control that triggered the event
- MC-EVENT → The name of the event (e.g., "Clicked")

This keeps it simple: you always know who did it and what they did.

5. Methods

Methods are like giving your controls little superpowers — making them do something on command. You can tell a control to refresh, animate, scroll, or do other tricks.

If you care about the result (e.g., a method that returns data), use DOMETHOD. If you don't, use DOMETHODNORET (it's even faster!).

6. Data Templates

Want to show a list of things, like products or contacts? A **Data Template** tells the app *how each item should look*. This powerful feature let's you represent each item with a group of controls instead of just a piece of text or picture.

In Mobile COBOL, you can build a Data Template step-by-step (CREATETEMPLATE / ENDTEMPLATE) and bind fields from your COBOL data to UI controls.

7. Sequential vs. Targeted Additions

Normally, when you add a control, it just goes to the current container on top of the stack. But sometimes, you want to be precise. In those situations you can use a TargetName property to say *exactly* where something should go.

8. The Shape of an API Call

Every API call uses **name/value pairs** separated by ; and ends with a | character. This | is like the period at the end of your sentence — it tells the DLL, "I'm done talking now."

Example:

```
CALL MC USING ADDTOCONTAINER "Name=MainStack;Type=Label;Text=Hello|".
```

If you remember these eight ideas, you'll have the Mobile COBOL “mental model” nailed down — and the rest of the guide will make perfect sense.

What's Next

We're just getting warmed up, but this is probably a great time to get your hands dirty by running one of the sample programs included with Mobile COBOL.

Running Your First Mobile COBOL Sample

You've read the guide, you've met the core concepts, and now it's time for the fun part — actually running a Mobile COBOL app. We'll walk through running one of the sample programs we ship with Mobile COBOL. By the end, you'll know exactly how to go from “hmm, what's this?” to “wow, I built that!”

Step 1: Locate the Samples

When you installed Mobile COBOL, we tucked a folder of sample programs into your installation directory. Look for a folder called:

MobileCOBOL\Demo

Inside, you'll find several ready-to-run programs — each one designed to show off a different feature. For this walkthrough, we'll use **HelloMobile**.

Step 2: Understand the Two Sides

Every Mobile COBOL app has two pieces:

1. **The Server Program** — This is your COBOL code running on Windows. It uses the Mobile COBOL API to build UI and respond to events.
2. **The Mobile Client** — This is our generic .NET MAUI app that connects to the server and renders the UI.

You can think of the server as “the brain” and the client as “the face.”

Step 3: Start the Server Program

1. Open your COBOL development environment (e.g., GnuCOBOL, Micro Focus, etc.).

2. Open the **HelloMobile.cbl** sample.
 3. Build and run it.
 4. Leave it running — it will sit there patiently, waiting for a client to connect.
-

Step 4: Connect with the Mobile Client

1. On your mobile device (or Windows machine), launch the **Mobile COBOL Client** app.
2. In the configuration screen:
 - **IP Address** → Enter the IP address of your server machine.
 - **Port** → Enter 5000 (or whatever you set in the sample).
3. Tap **Connect**.

If everything went well, the client will instantly render the UI described by the server's COBOL code.

Step 5: Interact & Observe

Click buttons, enter text, scroll lists — all the events you trigger are sent to your COBOL program, which responds in real time.

Try:

- Changing a label by clicking a button.
 - Navigating to a second page and back.
 - Watching how properties, events, and methods come together.
-

Step 6: Make It Your Own

Once you've run **HelloMobile** successfully, open the .cbl file and tweak it:

- Change some Text property values.
- Add a new button using ADDCHILD.

Run it again and watch your changes come to life.

Pro tip: All our samples are written to be short and approachable. Each one introduces just a couple of new ideas. Once you can run one, you can run them all — and more importantly, you can start making your own from scratch.

This guide showed you how to write your first Mobile COBOL app, but there's a whole world of UI controls, event handling, and power features waiting.

Next up: a full API reference with all the function calls, object types, properties, and examples you'll need.

But for now? Keep playing. Build a few screens. Try new controls. See what happens. It's COBOL — just mobile now.