

Mobile COBOL API Reference

This reference documents the Mobile COBOL API surface as of the current Windows-only release. Entries use consistent structure and name/value conventions.

Conventions

- Calls use name/value pairs separated by ';' and terminated by '|' (pipe).
 - Strings are literal; booleans accept True/False; numbers accept integers or decimals unless noted.
 - Target objects are referenced by their Name if required (e.g., Name=MyButton).
 - Control types, properties, methods, and events map to .NET MAUI equivalents when applicable.
 - Navigation is stack-based in this version.
-

Errors & Diagnostics

- DLL returns error messages via GETEVENT
-

Page & Navigation

CREATEPAGE

Purpose: Begin the definition of a new Page. The Name parameter is required so that the page can be referenced in other API calls, but additional properties can be set as well. Refer to .Net MAUI's `ContentPage` for additional properties.

Call form:

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

Required Parameters:

- Name=string (required) — Unique page identifier.

Behavior:

- Creates a new page definition.
 - Subsequent ADDCHILD and related calls add content to this page.
 - Does not display the page. Use SHOWPAGE to render.
-

CREATEPAGEXAML

Purpose: Create a complete page from a XAML file description.

Call form:

```
CALL MC USING CREATEPAGE "FileName=mypage.xaml|".
```

Required Parameters:

- FileName=string(required) – The file which contains the xaml page description.

Behavior:

- Creates a new page.
 - Does not display the page. Use SHOWPAGE to render.
-

ADDCHILD

Purpose: Add a control or container to the most recently referenced page or container, or to a specific page or container by using TargetName.

Call form:

```
CALL MC USING ADDCHILD "Type=Label;Text=HELLO;Name=Greeting|".
```

Parameters:

- Type=string (required) — .NET MAUI control type (e.g., StackLayout, Grid, Label, Button, Entry).
- TargetName=string (optional) — Name of the container the control should be added to.
- Name=string (optional) — Developer-assigned identifier for later reference.
- [properties]=various — Any supported property names for the Type (e.g., Text, Padding, Margin).

Behavior:

- If Type is a container (e.g., StackLayout, Grid), it becomes the active layout until ENDCONTAINER.

SHOWPAGE

Purpose: Display the current page on the mobile client.

Call form:

CALL MC USING SHOWPAGE.

Parameters:

- (none)

Behavior:

- Commits the pending page definition and pushes it on the navigation stack.
-

NAVBACK

Purpose: Navigate back one page in the stack.

Call form:

CALL MC USING NAVBACK.

Parameters:

- (none)

Behavior:

- Pops the current page from the navigation stack and shows the previous one.
-

Properties & Methods

SETPROP

Purpose: Set a property on a control.

Call form:

CALL MC USING SETPROP "Name=Greeting;Text=Welcome!|".

CALL MC USING SETPROP "Text=Welcome!|".

Parameters:

- Name=string (optional) — control Name. If this is omitted the last control which was referenced will be used as the target
- PropertyName=value (required) — where PropertyName is one of the property names associated with the referenced .Net MAUI control and value is the new value to be assigned to that property

Behavior:

- Applies the property change to the live UI.
- Type conversion follows MAUI conventions (e.g., colors, thickness).

GETPROP

Purpose: Retrieve a property value from a control.

Call form:

```
CALL MC USING GETPROP "Name=Greeting;Property=Text|" DATANAME.
```

Parameters:

- Name=string (optional) — Control Name. If this is omitted the last control which was referenced will be used as the target
- Property=string (required) — .NET MAUI property name.
- DATANAME (required) – COBOL data item that should receive the property value

Behavior:

- Returns the value of the property to the specified COBOL data item

DOMETHOD

Purpose: Invoke a method on a control and receive a return value.

Call form:

```
CALL MC USING DOMETHOD "Name=Entry1;Method=Focus|" DATANAME
```

```
CALL MC USING DOMETHOD "Name=List1;Method=ScrollTo;Arg0=10|" DATANAME
```

Parameters:

- Name=string (required) — control Name.

- Method=string (required) — .NET MAUI method name.
 - Arg0..ArgN=string (optional) — Positional arguments as strings.
 - DATANAME – COBOL data item that will receive the return value from the method.
-

DOMETHODNORET

Purpose: Invoke a method on a target control without expecting a return value.

Call form:

CALL MC USING DOMETHOD "Name=List1;Method=ScrollTo;Arg0=10|"

Parameters:

- Name=string (required) — Control Name.
- Method=string (required) — .NET MAUI method name.
- Arg0..ArgN=string (optional) — Positional arguments.

Behavior:

- Executes the method and does not return a value.
-

Layout, Templates, and Items

 Container Stack Rules (Quick):

- Adding a container control (e.g., StackLayout, Grid) pushes it onto the layout stack.
- ENDCONTAINER pops the top container off the stack.
- SHOWPAGE automatically closes any containers still open.

ENDCONTAINER

Purpose: Close the most recent open container layout.

Call form:

CALL MC USING ENDCONTAINER.

Parameters:

- none
- Pops the active container. New ADDCHILD calls add to the parent container.

Notes:

- Use to exit nested containers created by ADDCHILD for layouts.
-

ADDITEM

Purpose: Add an item to an item-based control (e.g., CollectionView).

Call form:

CALL MC USING ADDITEM "Type=ExpandoObject;Text=Alabama;Value=AL|".

CALL MC USING ADDITEM "Type=ExpandoObject;BindingName=Title;Value=Hello|".

Parameters:

- Type=ExpandoObject (required)
- PropertyName=value (optional) — Properties to set on newly created control

Behavior:

- Appends an item to the control's items source.
-

CREATETEMPLATE

Purpose: Begin a data template definition for templated controls.

Call form:

CALL MC USING CREATETEMPLATE.

Parameters:

- (none)

Behavior:

- This template is applied to the most recent Items control.
- Subsequent ADDCHILD calls define template visuals until ENDTEMPLATE.

Notes:

- Use with Items controls like CollectionView.
-

CREATETEMPLATEFOR

Purpose: Begin a data template definition for templated controls and apply it to the specified Items control.

Call form:

CALL MC USING CREATETEMPLATE "TargetName=List1|".

Parameters:

- TargetName=string (required) — Target control name (must support items).

Behavior:

- This template is applied to the Items control specified by TargetName.
- Subsequent ADDCHILD calls define template visuals until ENDTEMPLATE.

Notes:

- Use with Items controls like CollectionView.
-

ENDTEMPLATE

Purpose: End the current template definition.

Call form:

CALL MC USING ENDTEMPLATE.

Parameters:

- (none)

Behavior:

- Closes the template definition started by STARTTEMPLATE.
-

Events, Files, and Alerts

GETEVENT

Purpose: Retrieve the next pending UI event/message from the client.

Call form:

CALL MC USING GETEVENT MC-EVENT-INFO

Parameters:

- MC-EVENT-INFO (required) – Defined in MobileCOBOL.WS and stores the name of the event and the name of the control that triggered the event.

Behavior:

- Blocks until an event is available.

UPLOADFILE

Purpose: Upload a file from server to client.

Call form:

CALL MC USING UPLOADFILE "FileName=photo.jpg|".

Parameters:

- FileName=string (required) — name of file to upload.

DISPLAYALERT

Purpose: Display a simple alert/dialog on the client.

Call form:

CALL MC USING DISPLAYALERT "Title" "Body Text" "Button Label".

Parameters:

- First argument (required) – Title to display in alert box
- Second argument (required) – Text of the body of the alert box
- Third argument (required) – Label to display on the button in the alert box

Behavior:

- Shows a modal alert.

Data Templates — Detailed Guide

Data templates define how each item in a list-like control (e.g., `CollectionView`) is displayed. The control takes the data template and duplicates it for every control in the list. You set the properties of the controls in the data template by specifying bindings on your items. Let's take a look at how this works.

Creating Items with `ExpandoObjects`

The first step is to add the items to your list. This is done with the `ADDITEM` API. Use `ADDITEM` with `Type=ExpandoObject` to append a data object to the `Items` collection of a target list control. The `ExpandoObject` is a special object that will let you specify the properties you want to use in the template. Each name/value pair after `Type` becomes a property name and value for that item. You can pick whatever property names you want, but reserved keys are `Type`, `Name`, and `Template` so avoid those.

Call form:

```
CALL MC USING ADDITEM "Name=Orders;Type=ExpandoObject;Value=1001;Desc=Short  
info;ImagePath=/images/item1.png|".
```

Parameters:

- `Name=string` (optional) — Control Name.
- `Type=ExpandoObject` (required).
- `[properties]` — Arbitrary name/value pairs that become `ExpandoObject` properties.

Defining the Template

`CREATETEMPLATE` or `CREATETEMPLATEFOR` begins the visual definition for each item in the target items control.

Call form:

```
CALL MC USING CREATETEMPLATE.
```

Build the template using `ADDCHILD`, `SETPROP`, and other calls as if defining a normal layout, then close it with `ENDTEMPLATE`. Bindings map control properties to `ExpandoObject` properties using the syntax `Binding=<TargetProperty> to <SourceProperty>`. The `TargetProperty` is the property of the UI control that will be set and `SourceProperty` is the name of the `ExpandoObject` property that will be used to get the value.

Example: Template for Orders CollectionView

CALL MC USING STARTTEMPLATE .

CALL MC USING ADDTOPAGE "Type=VerticalStackLayout|".

CALL MC USING SETPROP "Padding=10|".

CALL MC USING ADDTOPAGE "Type=Label;FontSize=20|".

CALL MC USING SETPROP "Binding=Text to Desc|".

CALL MC USING ADDTOPAGE "Type=Image|".

CALL MC USING SETPROP "Binding=Source to ImagePath|".

CALL MC USING ENDCONTAINER.

CALL MC USING ENDTEMPLATE.