By default, Kentico uses the *HttpBrowserCapabilities* class of the .NET Framework to detect properties of the current visitor's device. However, this detection method has limitations and might not be sufficiently accurate for your needs.

If you want to utilize more accurate device detection and a wider range of device properties, we recommend integrating 51Degrees device detection.

Integrating 51Degrees

1. Open your Kentico solution in Visual Studio.
2. Right-click the Kentico web project (CMS or CMSApp) in the **Solution Explorer** and select **Manage NuGet Packages**.
3. Install the 51Degrees.mobi-web NuGet package.
4. Make sure that the imageOptimisation and redirect features are not enabled in the *~/51Degrees.config* file.
5. (Optional) Register the premium data file by modifying a detection key in the *fiftyOne* section of the *~/51Degrees.config* file. Modify the path to point to the location of your premium data file.

Example

```
<detection enabled="true" autoUpdate="true" binaryFilePath="~/App_Data/51Degrees.dat" />
```

Premium Device Data Subscription

You need to purchase a premium license from the 51Degrees website to gain access to the full list of properties and devices. By default, only properties labeled as **Lite** in the 51Degrees Property Dictionary are available.

Customizing CurrentDevice object properties

If you want to use the 51Degrees properties in macros directly as properties of the *CurrentDevice* object in the *{% CurrentDevice.IsMobile %}* format, rather than the *Data* container format *{% CurrentDevice.Data["IsMobile"] %}*, you can implement a custom *CurrentDevice* instance. The customization also introduces the 51Degrees properties to the macro autocomplete and helps you when entering macros in the administration interface.

Start by preparing a separate project for custom classes in your Kentico solution:

1. Open your Kentico solution in Visual Studio.
2. Create a new *Class Library* project in the Kentico solution (or reuse an existing custom project).
3. Add references to the required Kentico libraries (DLLs) for the project:
   - Right-click the project and select **Add -> Reference**.
   - Select the **Browse** tab of the **Reference manager** dialog, click **Browse** and navigate to the ***Lib*** folder of your Kentico web project.
   - Add references to the following libraries (and any others that you use in the custom code):
     - *CMS.Base.dll*
     - *CMS.Core.dll*
     - *CMS.DeviceProfiles.dll*
     - *CMS.Helpers.dll*
4. Reference the custom project from the Kentico web project *(CMSApp* or *CMS)*.
5. Edit the project's **AssemblyInfo.cs** file (in the *Properties* folder).
6. Add the *AssemblyDiscoverable* assembly attribute:

AssemblyInfo.cs

```
using CMS;

[assembly:AssemblyDiscoverable]
```

Continue by creating classes that register the required *CurrentDevice* properties:

1. Create a new class file under the custom project:
    a. Right-click the project and select **Add -> Class**.
    b. Enter *FiftyOneCurrentDevice.cs* as the file name.
    c. Click **Add**.
    d. Make the class inherit from **CurrentDevice** (found in the *CMS.DeviceProfiles* namespace).
    e. Create wrappers for any 51Degrees properties that you wish to access in your *CurrentDevice* macros:

> Each wrapper consists of a class property registered by the **RegisterColumn** attribute. Load the property values from the **Data** container, which is available in the *CurrentDevice* base class. Installing the 51Degrees NuGet package automatically ensures that the corresponding properties are available in the *Data* container.

Example - FiftyOneCurrentDevice.cs

```
using CMS;
using CMS.DeviceProfiles;
using CMS.Helpers;

public class FiftyOneCurrentDevice : CurrentDevice
{
    [RegisterColumn]
    public bool IsMobile => ValidationHelper.GetBoolean(Data["IsMobile"], false);

    [RegisterColumn]
        public string PlatformName => ValidationHelper.GetString(Data
["PlatformName"], "");

        ...
}
```

2. Create a new class file named *FiftyOneCurrentDeviceProvider.cs*.
    a. Make the class implement the **ICurrentDeviceProvider** interface (found in the *CMS.DeviceProfiles* namespace).
    b. Add the **GetCurrentDevice()** method and return an instance of your custom class containing *CurrentDevice* properties (*FiftyOneCurrentDevice* in this example).
    c. Register the class as the implementation of the *ICurrentDeviceProvider* interface using the **RegisterImplementation** assembly attribute.

Example - FiftyOneCurrentProvider.cs

```
using CMS;
using CMS.DeviceProfiles;

[assembly: RegisterImplementation(typeof(ICurrentDeviceProvider), typeof
(FiftyOneCurrentDeviceProvider), Lifestyle = CMS.Core.Lifestyle.Transient)]

public class FiftyOneCurrentDeviceProvider : ICurrentDeviceProvider
{
    public CurrentDevice GetCurrentDevice()
    {
        return new FiftyOneCurrentDevice();
    }
}
```

3. **Build** the custom project.

You can now access the registered properties within the *CurrentDevice* macro object.