Mobile Direct Participant Monitoring System Programmer's Guide

U.S. Department of Labor

Bureau of International Labor Affairs

September 17, 2018

IMPAQ INTERNATIONAL LLC

SUBMITTED TO

U.S. Department of Labor
Bureau of International Labor Affairs

ATTENTION

U.S. Department of Labor
Bureau of International Labor Affairs, Room S-5317
200 Constitution Ave. NW
Washington, DC 20210
United States

SUBMITTED BY

IMPAQ International, LLC
10420 Little Patuxent Parkway
Suite 300
Columbia, MD 21044
(443)256-5500
www.impaqint.com

CHILD LABOR DIRECT PARTICIPANT MONITORING SYSTEM

Mobile Direct Participant Monitoring System Programmer's Guide
Funding is provided by the United States Department of Labor under cooperative agreement number IL-26684-14-75-K-24.

Table of Contents

1. Overview of the mDPMS

The Mobile Direct Participant Monitoring System (mDPMS) Programmers Guide serves as a high-level overview for software developers who will be developing, modifying, or maintaining the mDPMS. For an overview of mDPMS application features, please see the mDPMS User Guide. The mDPMS was created for Android using Xamarin Forms and C#. The system is capable of being developed on either a Windows or macOS environment.

If changes (code modifications) are made to the DPMS it may be necessary for the software developer to make changes to the mDPMS.

Possible reasons for changes include (but are not limited to):

- Modifying the DPMS API
- Changing the DPMS models or other schema changes affecting the API
- Adding or removing a feature in the DPMS
- Modifying the standard forms in the DPMS

It is the responsibility of the developer to test and determine what changes are needed and to ensure that the changes are implemented correctly.

2. Skills needed by Contributing Developers

The following is a list of technologies used in this project. It would be helpful for contributing developers to have experience with the following technologies.

- Xamarin Forms
- Android
- C#
- .NET
  - Core
  - Standard
- Entity Framework
  - Core
- REST
- JSON

3. Development Environment Set Up

This section provides instructions on how to set up a development environment to assist in the building, modification, or maintenance of the mDPMS application. These instructions are subject to change as outside applications are updated. As always, software developers should use their best judgement when setting up an environment.

The following is an environment for Windows based PCs. All software requirements are available free of cost except for a Google Play Store account for distribution. Ad Hoc distribution is a cost-free alternative method of distribution.

Below is a list of development environment prerequisites:

- Windows 7 or newer computer with administrator permissions
- Internet access
- Android device or computer capable of running an emulator

Below is a list of optional development environment components:

- Multiple Android devices
- Processor supporting HAXM if emulators will be used

Below are instructions on installation of the development environment:

- Install Visual Studio Community Edition
  - Download from https://www.visualstudio.com/
  - Install with the following options:
    - Install the Workload "Mobile Development with .NET"
      - Add the optional component "Universal Windows Platform tools for Xamarin" in the workload if you plan to add a UWP target
    - Install the following from "Individual Components"
      - ".NET Core Runtime"
      - "Git for Windows" (optional but recommended)
      - "GitHub extension for Visual Studio" (optional)
      - "Powershell tools" (optional)

        - The total size will be approximately 14.74 GB
        - The install could take some time

- Set up your preferred Git tools/preferences and download the code from GitHub

- Set up a device or emulator for previewing and debugging
    - To set up an emulator
        - Enabling HAXM is recommended (see https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-emulator/hardware-acceleration?tabs=vswin for more detailed information)
    - To use physical device or devices
        - Use your preferred Android device (see https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/set-up-device-for-development for further guidance)

4. mDPMS Code Overview

This section details the Xamarin solution provided in the source code. It also covers some important concepts where appropriate.

- MDPMSAssets
  - This is a folder containing assets needed for distribution and also assets helpful for development.
  - The subfolders “Art” and “PlayStoreArt” contain art assets
  - “AddMigration.ps1” and “AddLocalizationMigration.ps1” are PowerShell scripts that can be used to create database migrations for versions for the application. Bash versions are also available under the same file names with .sh extensions.
  - “export_tables.sh” and “compare_files.sh” are Bash scripts that can be used during development to export and compare database states.
- MDPMS.Android
  - This is a project generated by creating a Xamarin Forms project. It is the only target in this application. Minimal modifications should be needed here. Generally only reference additions, initialization changes, and art asset additions are needed.
- MDPMS.Database.Data
  - This project is a .NET Standard 2.0.1 Library using Entity Framework Core 2.0.1 to create the database context and data models used in an mDPMS instance.
  - “Database” Folder
    - “DatabaseSeed.cs” contains a static method to seed a database instance with genders. This method can also be used to seed other tables if needed.
    - “MDPMSDatabaseContext.cs” defines the DbContext for a database instance.
  - “Migrations” Folder
    - This folder is in place to be used with the migration PowerShell or Bash scripts to generate database migrations.
  - “Models” Folder
    - This folder defines the data models used in the database context.
    - “Base.EfBaseModel.cs” provides a base class for use in models that are synced with the DPMS.
      - Important concepts:
        - Internal vs. External Id

        - Internal Ids will always exist on object creation.
        - External Ids should only exist on a synced object. This external Id should be the id that the parent DPMS generates.
      - Created and Updated At DateTimes
        - All syncing is based on updated_at property from the parent to the local property LastUpdatedAt of each object.
      - SoftDeleted exists for later use in more complex editing and syncing requirements.
    - "Base.ISyncModel.cs" provides interfaces to be implemented in data models intended to be used in the syncing code.
    - The rest of the models define an object used in the local database context.
- MDPMS.Database.Localization
  - This project defines the database context for a second database in the application used for providing localizations wherever non-data text is displayed. This is similar to "MDPMS.Database.Data" except that this seeds a localization database from a translations list.
  - The database structure is very simple in that it contains only definitions for localizations, keys, and values based on a key + localization.
- MDPMS.Helper.Gps
  - This is a .NET Standard 2.0.1 class library used as a wrapper around the Xam.Plugin.Geolocater Nuget. All helper class libraries are intended to provide simple wrappers around different pieces of functionality.
- MDPMS.Helper.Json
  - This is a .NET Standard 2.0.1 class library used as a wrapper around the Newtonsoft.Json Nuget. This is used to simplify reading to and writing from the local AppData.json file.
- MDPMS.Rest
  - This is a .NET Standard 2.0.1 class library used as a wrapper around the RestSharp Nuget. This is used to simplify making REST-ful requests from the parent application's API.
- MDPMS.Shared

- This project is the main Xamarin Forms application referenced by the Android target. This is meant to be cross platform code which can be used by other supported targets. This project is also .NET Standard 2.0.1 compatible.
- "Converters" Folder
  - This currently contains only 1 Converter, a ReverseBooleanConverter which simply flips the value of a boolean to aid in enabling and hiding content on views. This folder can also contain other Converters used in view value conversions.
- "Models" Folder
  - This folder is intended to contain model definitions not related to database model definitions. The Model in Model-View-ViewModel is split between this folder, MDPMS.Database.Data, and MDPMS.Database.Localization.
  - ApplicationInstanceData.cs defines instance level application data intended to be initialized at start up and passed between views. This contains the database contexts and other useful application instance level data.
  - Localization.cs defines a model to cache localizations from the localization database in memory. ApplicationInstanceData.cs contains a variable SelectedLocalization which is the currently selected localization to be referenced by views.
  - SerializedApplicationInstanceData.cs defines the model for data that will be persisted in the AppData.json file for an application instance.
- Views and ViewModels Folders
  - Each view in the application has both a view and a corresponding view model. There are 2 types of views in mDPMS, ContentViews and ContentPages. ContentViews are reusable content shown in ContentPages. All views are XAML.
- Workers Folder
  - This contains SyncWorker.cs which runs all of the actions related to syncing the local database with the parent.
- App.xaml contains the styles used in the views.
- App.xaml.cs is used to initialize the application and the instance of ApplicationInstanceData which will be passed between views.
- RootPage.xaml and RootPage.cs create the MasterDetailPage used as the basis for presenting all views.

5. mDPMS Deployment

See https://docs.microsoft.com/en-us/xamarin/android/deploy-test/publishing/ for more information on deploying the application to users.

Two good options are:

1. Google Play Store (see https://docs.microsoft.com/en-us/xamarin/android/deploy-test/publishing/publishing-to-google-play/?tabs=vswin)
2. Ad Hoc Distribution (see https://docs.microsoft.com/en-us/xamarin/android/deploy-test/publishing/publishing-independently)