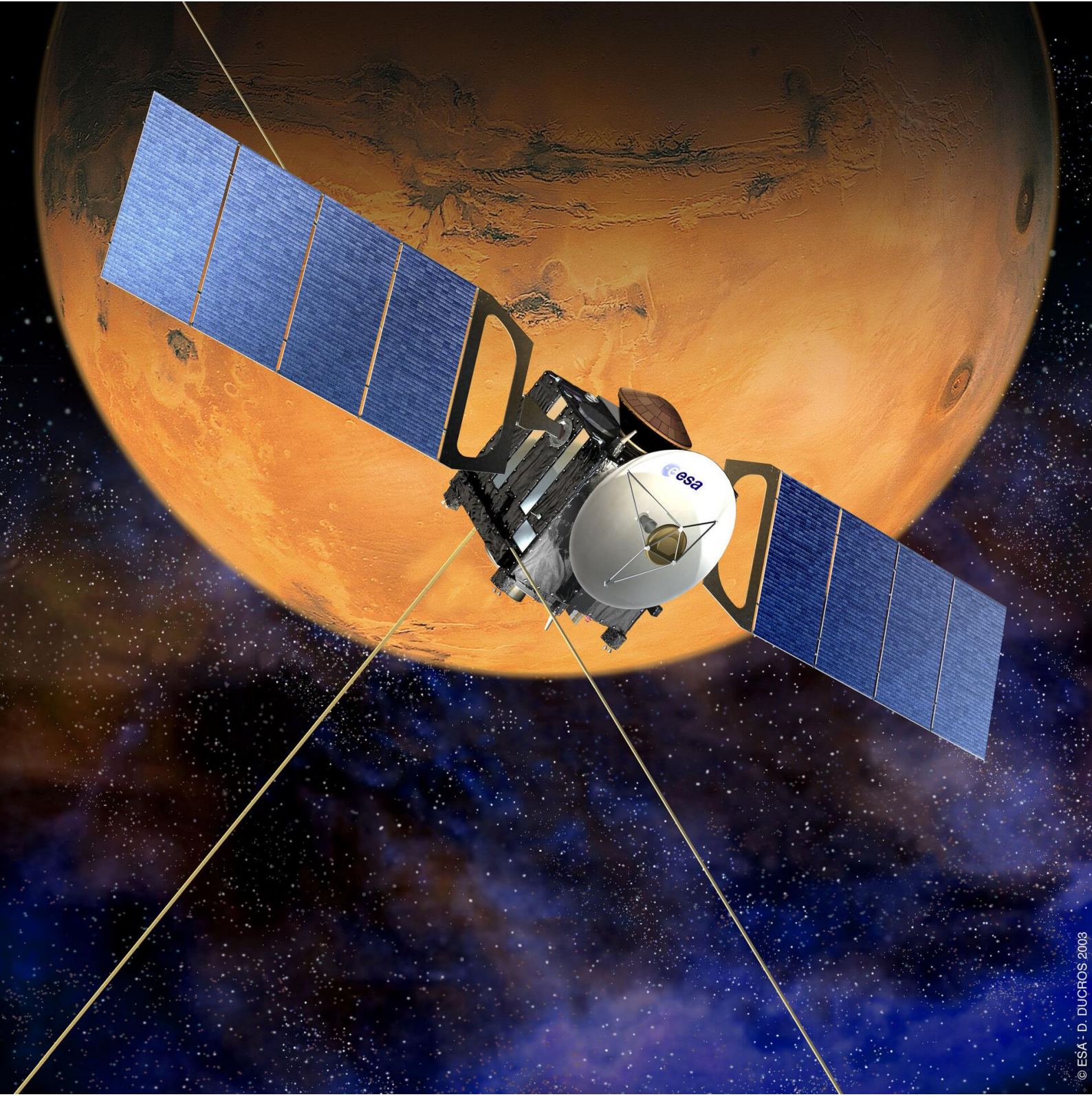

PFS Data Manager User Manual

Francesco Carraro

1) Overview

1.1) What is PFS Data Manager?

The *PFS Data Manager* is a tool developed to managing the process of creation and sending of requests for retrieving data related to the PFS instrument, on board the *Mars Express* mission. The app also allows the processing of the retrieved files.

In this user manual it will be explained how to use the application for sending requests and process retrieved files.

1.2) INAF DOI

The software has been registered with the following DOI: INAF.SW.2022_00003

1.3) Software requirements

PFS Data Manager is a UWP (Universal Windows Platform) application which can only run on Windows 10/11. It is designed for being used with a mouse but also supports touch interactions, if runs on a touchscreen equipped computer.

2) Installation

Installation and update of *PFS Data Manager* is pretty simple thanks to the ms-appinstaller protocol that has been realized by Microsoft to let developers creating the so-called LOB (Line Of Business) apps: apps developed for internal usage by companies that are not interested in publishing their apps on the Microsoft App Store.

When creating the app package in Visual Studio, an option is present to set an URL used by the app itself to search for an update when running. If this option is selected, an *index.html* file and a file with extension *.appinstaller* is created together with the app package.

Following the web page is shown.

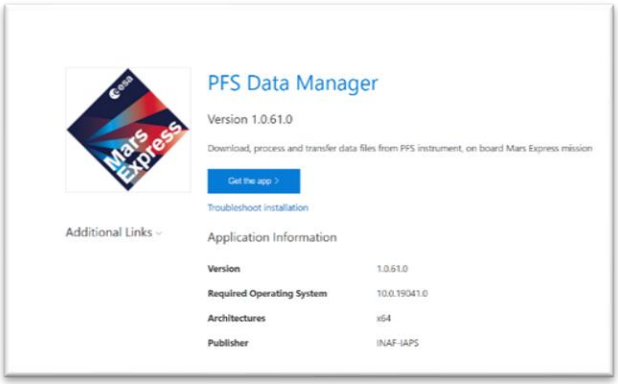

Figure 1 The web page for app installation

By simply clicking on the *Get the app* button, a panel is shown, on top of the page, asking the user the authorization for installing the app.

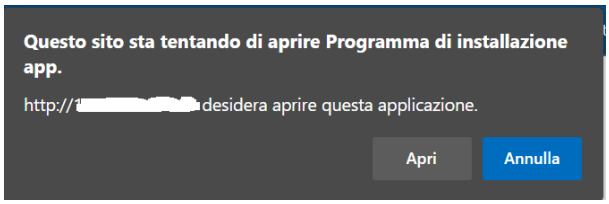

Figure 2 Alert asking user permission for starting the installation process

Once pressed the *Open* button, the install process starts.

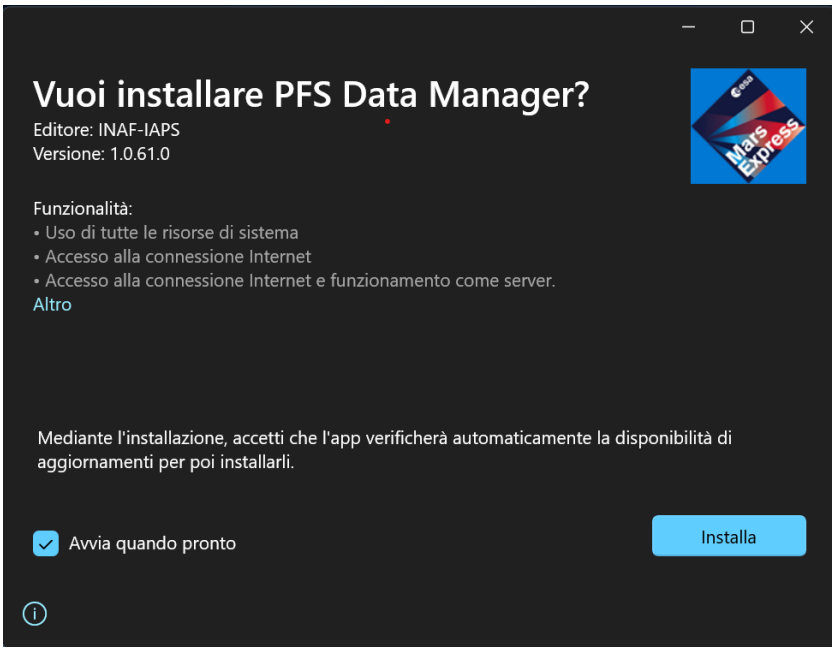

Figure 3 UI for starting installation process

The UI for starting the installation process shows a few info about permissions required by the app for working and the option for starting the app once the installation is completed.

NB. Both the panel for authorization and the UI for installation are automatically localized by using the OS language.

2.1) Pre-installation

2.1.1) Enabling the ms-appinstaller protocol

The installation of apps following the method described in the earlier paragraph, called side-loading, is usually disabled on Windows since Microsoft discovered that a few viruses were propagated by using it. In order to enable this protocol, a few changes are required in the Windows registry.

2.1.2) Importing the certificate

Each UWP app installed in Windows needs a certificate which grants for the developer and the origin of the app itself. The OS must know the certificate used by the app in order to allow the installation. If the OS doesn't know it, the *Install* button in the UI shown in *Figure 3* is disabled and a warning is shown.

Usually, when apps are published on the Microsoft store, they are signed by Microsoft during the validation process. So, the apps contains a trusted certificate that allows the OS to run them.

When using the side-loading system, apps are not signed by Microsoft and a not-so-simple procedure is required to import the certificate, before trying to install the app, by using a Windows utility.

2.1.3) Pre-installation app

To **simplify the installation process** and avoid each user to waste a certain amount of time in changing the Windows registry and importing the certificate, a simple console application has been realized which does all the work.

By running the application, the OS is made ready for installing apps by using the side-loading system. This step is needed only one time, before the first installation of the app.

2.2) Updates

One of the greatest benefits of using the ms-appinstaller protocol is the easiness of the installation process, once the OS is ready for the protocol.

Another great advantage of the ms-appinstaller is the management of the software updates for the app users: each time the app is closed, it silently searches for an update at the URL specified when the app package has been created. If an updated version of the app is found, then it is silently downloaded and installed. The next time the user opens the app, the latest version is available.

3) Usage

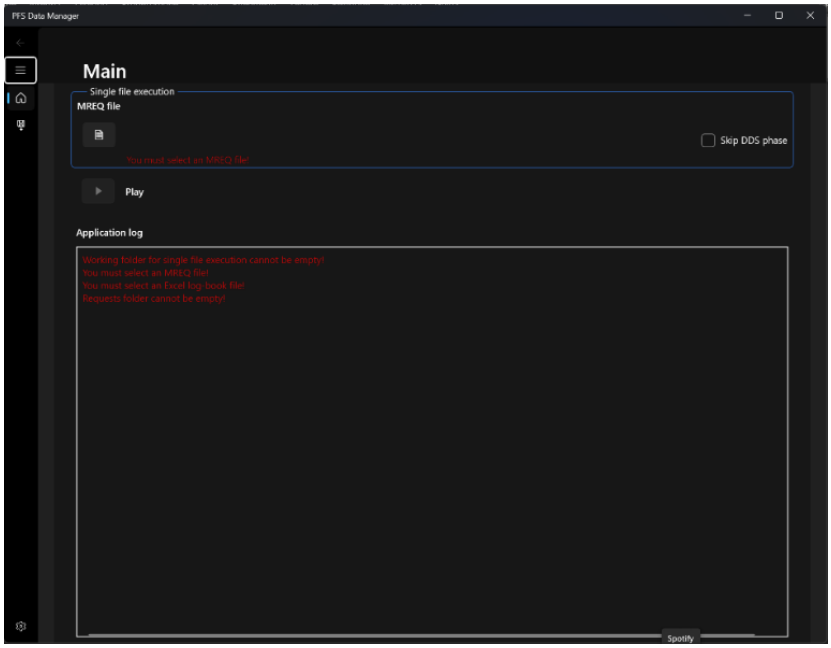

Figure 4 Home page

The first time the app is opened, a few settings need the user attention:

- Working folder: the folder containing files used as input for creating requests;
- Excel logbook: the Excel file which is filled at the end of the data processing with overall info about each orbit;
- Requests folder: is the folder where requests are created to be sent towards the *ESOC* server.

Settings page can be accessed by tapping on the settings icon in the bottom-left of the app. This action is required in order to start using the app.

3.1) Settings page

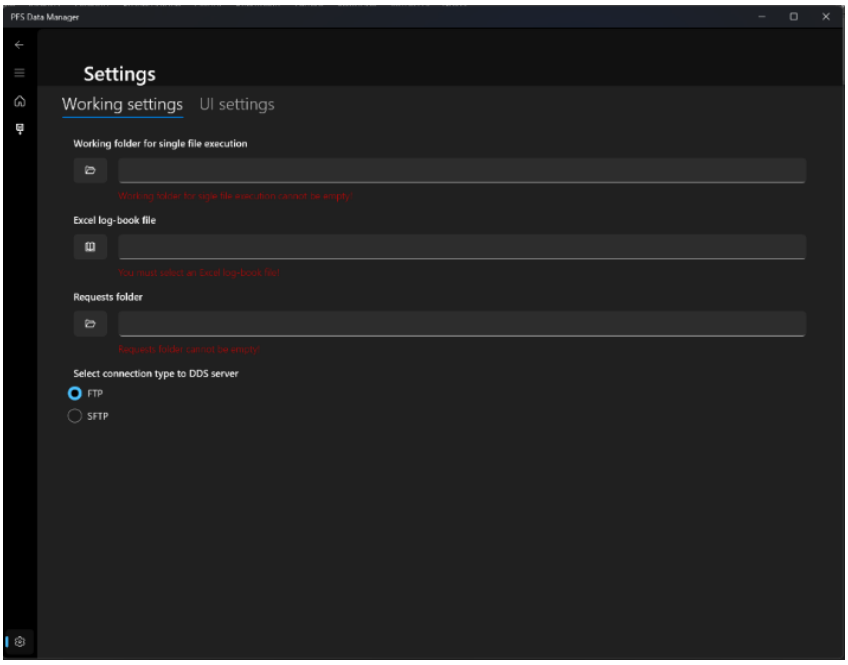

Figure 5 Settings page

This page is basically a one-shot page, which is typically used during the first app usage for defining the values of the required settings.

3.1.1) Working settings

The textboxes shown in *Figure 5* are the settings described in the previous paragraph. An option is also present at the bottom of the page in order to select the connection protocol for sending requests to the *ESOC* server.

3.1.2) UI settings

This section has settings related, as stated by the name, to the UI of the app. It means the theme used: light or dark. The light theme shows a white background for panels with black text; the dark theme is the opposite.

3.2) Data import

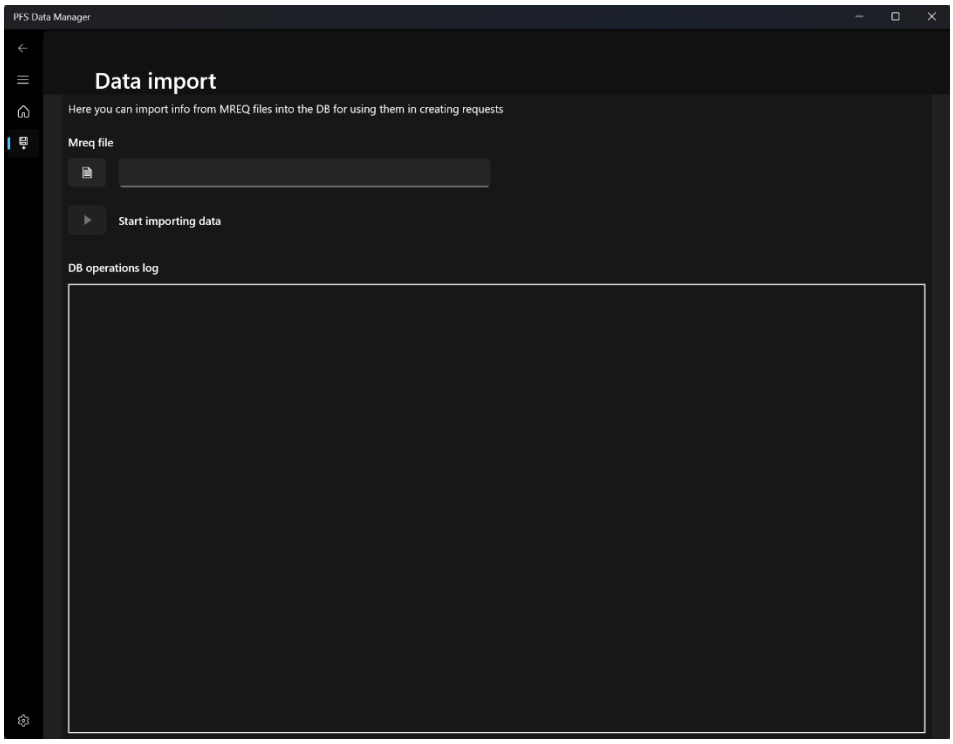

Figure 6 Data import page

Before starting the creation of requests, a second step is required: importing the complete set of information included in the file that contains the summary of all the orbit observations scheduled for each instrument.

Once selected the huge file containing the above mentioned information, the app starts reading each line, detecting the ones related to the *PFS* instrument.

When a line related to *PFS* is found, the content of the line is parsed and saved into a SQL Server database that has been developed for being used as a repository of data related to *PFS* observations. Those data will be then used for creating requests sent to *ESOC* server for retrieving *PFS* observations.

This operation is strictly required the first time the whole project is used. Uninstalling the app doesn't affect the database, which has an autonomous life. When the app is re-installed, it is automatically connected with the already existing database and no action is needed.

An eventually available update of the file holding the scheduled observations implies a new run of this feature, of course.

In order to start the reading process, use the *Mreq fil*e button to select the above-mentioned file containing all observations scheduled for *Mars Express* instruments.

Once the file has been selected, just press the *Play* button to begin the operation. A log window, as shown in *Figure 6*, is present, which is filled with lines related to *PFS* found during the reading process.

3.3) Managing data

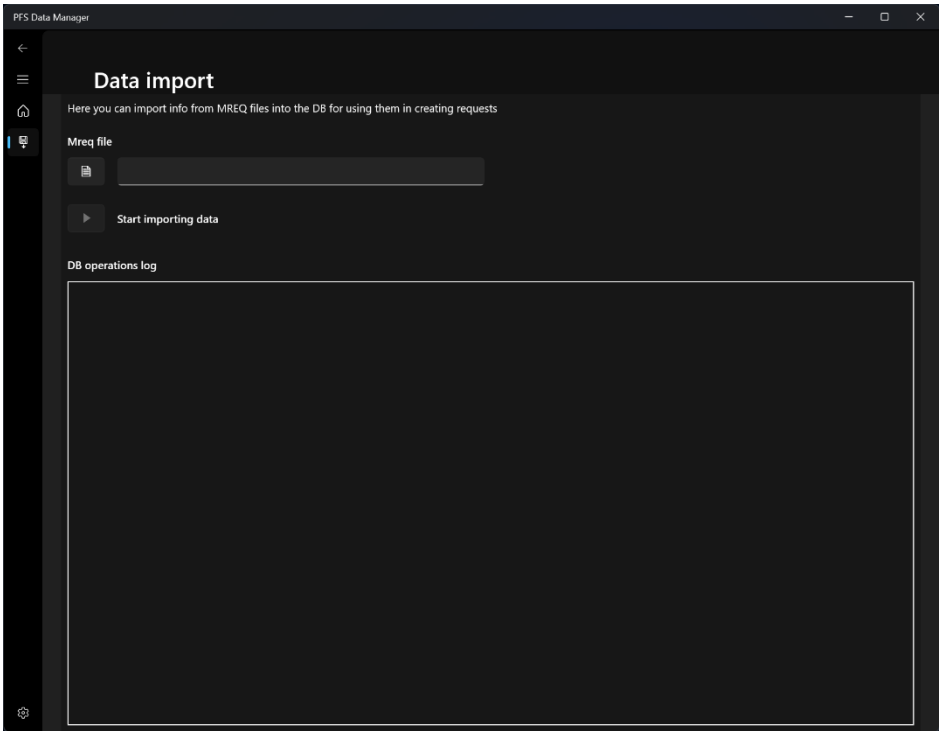

Figure 7 Home page ready for creating requests

When app is started, if all required values in settings page have been entered, the home page looks like the one shown in Figure 7.

The UI of the page is pretty simple:

- a *Mreq file* button for selecting the file used for creating requests;
- a *Play* button for starting process;
- a big window filled with operations log.

Usage of this page is the same of the *Data import* page: just select an input file by using the *Mreq file* button and press the *Play* button. No actions are required by user who has just to wait for the end of the process.

4) Retrieving data

Following a brief explanation of what happens *behind the curtains* when the process of creation of request and retrieving of data is started.

4.1) Creating requests

The first phase when the data retrieval process is started, is the creation of the set of requests that will be sent to the *ESOC* server.

Requests are XML files created starting by the file selected by user, as explained previously, and by using some data contained in the *SQL Server database*.

Each .MEX file holds data related to a set of orbits around *Mars* planet. Not all orbits have observation executed by *PFS*, so only the right ones are used to create a request.

Each observation produces an XML file which is sent to *ESOC* server. When all requests have been sent, the app starts waiting for dds files received from *ESOC* as responses for requests.

To each request corresponds a response, in the form of a dds file and no new actions are started until all expected responses have been received.

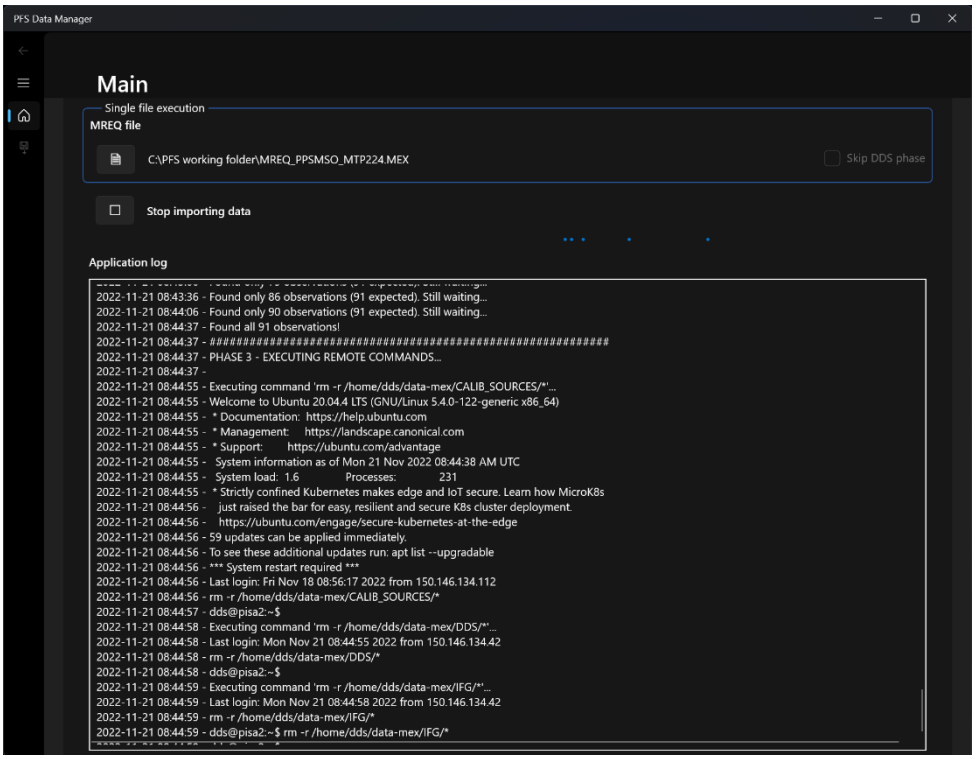

Figure 8 Home page during usage. On top of the log, lines related to the waiting for all responses are shown. Following lines related to requests processing are present.

4.2) Processing data

Once all requests have been processed and responses have been received from *ESOC*, a series processing phases starts.

4.2.1) Remote commands

The first processing phase consists in executing a few scripts for extracting data from dds files. Several science files are extracted from telemetries contained in dds files and also a complete log of each step is recorded.

4.2.2) Comparing requests and results

The second phase consists in a comparison between expected data contained in requests and retrieved results read from responses. The complete check result is logged into a text file.

4.2.3) Deleting 0 bytes files

It may happen that, for several reasons, observations may be not completed as expected or errors may occur during steps between the acquisitions of data by the instrument and the receiving of data by the ground segment. To avoid filling storage disks with 0 bytes files, a check of the so-called Mars extracted data is completed and all empty files are removed.

4.2.4) Moving data to storage

The last phase of the process is the moving of science files from the working directory to the final destinations, to be available for the science team.