DICOM Worklist System

MIELE: **M**edical **I**maging **ELE**ments

wl-feeder version 0.1

Table of Contents

1 General

The purpose of this project is to convert various types of input files into DICOM work-list items that can be either stored locally as DICOM files (they contain no image data, only work-list information), or sent across the network using the DICOM protocol. In either case the ultimate goal is to make the datasets available to a work-list server (example: `wlmscpfs`) so it will be able to reply to DICOM queries based on them.

The architecture of the application is based on plugins, each one able to understand and parse a different input file type. There is a small number of built-in plugins, but the functionality can be expanded by adding custom plugins (contact the author for details on how to achieve that).

Each “parsing” plugin is associated to a filename extension.

Modes of operation for the **input** files:

- either discovered by periodically polling a designated local input directory. After processing the file is deleted.
- or they can be sent to this application from a remote computer via HTTP-POST.
- both of the above modes of operation can also be used at the same time.

This application includes a HTTP-POST server that can be used to receive files via a network interface, for example using the command line utility `curl` or HTML forms, served by a web server.

The application toolbar has two buttons labelled `Start` and `Stop` which control the timer that polls the input directory every few seconds.,These two buttons only affect the “parsing” plugins, which will be notified of their own associated file.

The application assumes unattended operation so it doesn’t prompt for data input if missing, instead it will use default values specified in the GUI and in the Preferences.

Modes of operation for the **output** items:

- either store the work-list items locally as DICOM files with filename extension .wl
- or send them across the network via DICOM acting as C-STORE_SCU.
- only one of these two modes of operation can be active at any time.

2 Plugins and Input Files

Each plugin is represented by a “tab” in the user interface. The plugins are discovered dynamically when the application is launched.

2.1 .hl7

This built in “parsing” plugin handles HL7 files version 2.

Note that HL7 version 3 files are based on XML and a different plugin is required for them.

2.2 .gdt

This built-in “parsing” plugin handles .GDT files. GDT stands for *Geräte-Daten-Träger* (Device data volume) a file format mostly used in Germany and Switzerland.

Note that only GDT records of type `(PVS to DEVICE)` are supported.

If the GDT file doesn’t contain enough information to create an SPS, a default SPS is created using the GUI values as template and the current time as timestamp.

2.3 manual

This is not a “parsing” plugin: it is not associated with any filename extension.

It operates independently of the `Start` and `Stop` toolbar buttons. It is used to create work-list items from scratch, interactively, using the controls provided in the user interface.

Please make sure that you have edited the default SPS values before you click the `Create` button. Currently the GUI allows you to create additional SPSs but only the first one will be used.

3 Output items

3.1 Network operation

This application can send work-list items to a remote computer via the DICOM protocol, using the C-STORE service. This application acts as the (client) **SCU.** Obviously the counterpart (server) **SCP** must be running and listening to incoming associations.

A typical C-STORE **SCP** is the DCMTK utility `cstorescp` that can be launched like this:

```
$ cstorescp <port> --write-dataset --output-directory <mydir>
```

The directory `<mydir>` should be conveniently specified as the location where the WL server expects to find work-list items, typically the AET subdirectory matching the WL client that will then query the WL server.

4 Getting Started

1. Input settings

    1. define the local directory to be polled for incoming files. Note that after processing each file gets deleted.

2. Output settings

In the Preferences, “General” tab, “Output method” use the checkbox “Save to local directory” to select the behavior from the following two options:

1. *checked*: store the items as local files (define the directory). The resulting filename will be the timestamp of the operation with extension `.wl`.

2. *unchecked* send the items to a remote C-STORE-SCP. You must define the hostname and port of the SCP, and check that it is online with the `Verify` button. Typically the resulting filename on the remote PC depends on the SCP implementation, but typically it could be the `“SOP class UID”.”SOP instance UID”`

3. If planning to use the “parsing” plugins, click the `Start` button in the main toolbar

4. If planning to use the direct input WL item creation, click the `Manual` tab in the GUI. Note that one SPS is automatically added with placeholder values. Note also that the GUI allows creating more than one SPS but there is not guarantee that every WL server will be able to used that format. When ready click the `Create` button

5 System Diagram