DDOIF API Manual

Mohammed Al-Rawi
Adapt Centre, Trinity College Dublin
June 2, 2020

Entitled “Digital Data Organization and Exchange in Fashion (DDOIF)”, the project aims to develop an ICT standard seed that defines terminology, vocabulary, descriptions, categories, sub-categories, anatomy, characteristics, personal preferences, colors and patterns, fashion use-cases, as well as many other attributes related to fashion articles and related intelligent systems. Fashion companies that adopt DDOIF will gain a competitive market advantage as the standard will facilitate the development of powerful applications, to mention a few, brand forensics and protection, and the next-generation recommender systems that deliver a more effective and superior customer shopping experience.

DDOIF is pronounced this way /diːdɔɪf/

Introduction

The API is a software used to enable importing and exporting fashion items according to the DDOIF protocol. A DDOIF file has the following structure:

DDOIF File Header

We designed the DDOIF file to start with a 10-byte signature (ie Magic Number or Magic Bytes) containing the following:

{0x89} has the high bit set to detect systems that do not support one byte transmission data.

{0x44 '0x44' '0x4f' '0x49' '0x46'} in ASCII, the letters DDOIF enable identifying the format easily if it is viewed in a text editor.

{0x0D 0x0A} a DOS-style line ending (CRLF) to detect DOS-Unix line ending conversion of the data.

{0x1A} A byte that stops display of the file under DOS when the command type has been used—the end-of-file character.

{0x0A} A Unix-style line ending (LF) to detect Unix-DOS line ending conversion.

{Reserved bytes} 16 bytes are reserved for future additions, in case needed. Currently filled with 0 values.

Data Chunks

We designed DDOIF file in a manner that encapsulates textual data denoting fashion attributes, media types of a single piece of clothing and the other information according to the DDOIF dictionary. The textual data chunk consists of:

{4} bytes store the length in bytes (let the length be denoted by *N*) of the DDOIF textual description according to the DDOIF dictionary.

N bytes store the classes and attributes of the item according to the DDOIF dictionary.

It is well known that the design of fashion items may articulate different types of media data; for example, 3D objects, 2D images at different views (front, back, side, etc) and even videos. Each media format will constitute a data chunk in the DDOIF file. Consequently, we store the media data chunks sequentially, and each will contain the following stream of bytes:

{8} bytes store the format name (i.e. type) of the media, eg, JPG / JPEG , PNG, TIFF, STL, OBJ, 3DS, etc. This string will be needed to know the type of stored media and to decode it once it is read. We choose 8 bytes instead of 4 to enable future extension to formats that may have more than 4 letters.

{4} bytes to store the length of the Media Buffer, let it be denoted as *M*.

{*M*} bytes store the media buffer. We refer to the encoded media data as "Media Buffer". The Media Buffer is a stream of bytes that may be encoded into a compressed format.

4 bytes store the CRC (cyclic redundancy code/checksum. The CRC is a network-byte-order CRC-32 computed over the Media Buffer.

Whenever there are several media data, even of different formats, we store them in a sequential manner each having the aforementioned structure.

DDOIF filename extension

We add the ".ddof" extension to any DDOIF filename.

Functions

ddoif_io:ddoif_write(ddoif_dict, media_buffer, out_f)

A function used to write fashion textual item and the relevant mdedia as a ddof file.

Input arguments:

- ddoif_dict: dictionary containing textual description of clothing attributes according to ddoif dictionary
- media_buffer: a list of media buffer of the clothing item
- out_f: the name of output file that has .ddof extension

Output arguments:

- the_dict : python dictionary

Methods:

ddoif_write(ddoif_dict, media_buffer=media_buf, out_f='ATest.ddof')

ddoif_io:ddoif_read(in_f, check_CRC)

A function used to read fashion textual item and the relevant mdedia stored as a ddof file.

Input arguments:

- in_f: the name of output file that has ".ddof" extension
- check_CRC: A flag (True/False) whether to check the file integrity via CRC code

Output arguments:

- ddoif_dict: Textual attributes of the clothing item according to ddoif dictionayr
- media_buffer: a list of media buffer of the clothing item

Methods:

ddoif_dict = ddoif_read(in_f='ATest.ddof', check_CRC=True)

ddoif_io:dict_to_binary(the_dict)

A function used to convert a python dictionary to a byte string.

Input arguments:

- the_dict: python dictionary

Output arguments:

- dict_in_bytes : python dictionary as a byte string

Methods:

byte_dict = dict_to_binary(in_dict)

ddoif_io:binary_to_dict(dict_in_binary)

A function used to convert a python bytes to a dictionary.

Input arguments:
- dict_in_binary: python dictionary as bytes

Output arguments:
- the_dict : python dictionary

Methods:
the_dict = binary_to_dict(dict_in_binary)
returns python dictionary

ddoif_utils:read_yaml_as_dict(fname)

A function used to read the ddoif dictionary in yaml format and return it as a python dictionary. This functiona makes use of ruamel.yaml https://pypi.org/project/ruamel.yaml/

Input arguments:
- in_f: the name of yaml file

Output arguments:
- ddoif_dict: All attributes the ddoif dictionary

Methods:
ddoif_dict = ddoif_read(in_f='ATest.ddof', check_CRC=True)

ddoif_utils:yaml_to_xml(yaml_fname, xml_fname, ids=False)

A function used to read the ddoif dictionary in yaml format and save it into an xml file. This function makes use of dicttoxml package.

Input arguments:
- yaml_fname: the name of yaml file as input
- xml_fname: the name of output xml file

Output arguments:

Methods:
yaml_to_xml('ddoif_dictionary.yaml', 'ddoif_dictionary.xml')

ddoif_utils:yaml_to_xml(yaml_fname, xml_fname, ids=False)

A function used to read the ddoif dictionary in yaml format and save it into an xml file. This function makes use of dicttoxml package.

Input arguments:
- yaml_fname: the name of yaml file as input
- xml_fname: the name of output xml file

Output arguments:

Methods:
yaml_to_xml('ddoif_dictionary.yaml', 'ddoif_dictionary.xml')

ddoif_utils:yaml_to_json(yaml_fname, json_fname)

A function used to read the ddoif dictionary in yaml format and save it into a json file. This function makes use of json package.

Input arguments:
- yaml_fname: the name of yaml file as input
- json_fname: the name of ouput json file

Output arguments:

Methods:
yaml_to_xml('ddoif_dictionary.yaml', 'ddoif_dictionary.xml')

Example:

We used a set of images of a sport jacket at different views and conditions to populate a single DDOIF file. Via DDOIF API, we stored all these images into as single file in DDOIF format, along with classes defining attributes of the jacket taken from DDOIF dictionary. The file ***jacket.ddof*** is made publicly available at https://github.com/morawi/ddoif/tree/master/Examples and any user can use the DDOIF API to load it. We designed the file format in a way that accepts an unlimited number of media data.