MakeFmt

Make User Formats
User Guide

Version 1.0.1
18th March 2024

Paul Clarke

MQGem Software Limited
support@mqgem.com

Take Note!

Before using this User's Guide and the product it supports, be sure to read the general information under "Notices".

Third Edition, March 2024

This edition applies to Version 1.0.1 of *Make User Formats* and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

Table of Contents

Introduction

IBM MQ messages can be simple string messages, but often as not they are more complicated beasts than that. The MQEdit program allows you to define user formats to describe your messages so that when browsing and editing them, you can see the values of all the different fields.

Creating an MQEdit User Format file was previously something you had to do manually. MakeFmt is a free utility to make that process easier. It is capable of converting most COBOL copybooks to the structure format used by MQEdit.

My thanks go to Morag Hughson for considerable assistance with this manual, for numerous usability suggestions, for keeping me sane when nothing ever seems to go right, as well as tirelessly testing many buggy versions of the program!

If you have any questions about the product or perhaps suggestions for additional features then please send an email to support@mqgem.com.

Main changes from previous version

Unless otherwise stated the behaviour of the previous version should be maintained and, if all goes well, enhanced. While every effort is made to try and ensure that there are as few bugs as possible it would be surprising if some problems didn't leak out. We would recommend that users keep their previous version of ***makefmt*** handy so that they can revert to it if a bug is found. Needless to say, please report any incompatibilities to us if they are found and we will do our best to provide a fix.

The full list of changes are:

- Support added for COBOL `PIC S9(n) SIGN IS LEADING|TRAILING` being translated into the new MQEdit V9.3.3 User Format data type `opunch(n)`. See 3.1. Data Types on page 5.
- Support added for COBOL `PIC S9(n) SIGN IS LEADING|TRAILING SEPARATE` being translated into the new MQEdit V9.3.3 User Format data type `digits(n)`. See 3.1. Data Types on page 5.
- New options parameter, `-Os` added to allow reversion of the above back to the MQEdit User Format `hex` data type if you preferred viewing these types of fields in their raw format. See 4.2. Optional Parameters on page 9.

Chapter 1. Make User Formats

This document describes the functions available in the program.

1.1. Overview

The MQEdit program from MQGem Software will display your own messages in a formatted display if you provide the MQEdit program with a User Format file describing the shape of the message.

The *Make User Formats* (***makefmt***) program can be used to create an MQEdit User Format file from a COBOL copybook. Once created, it can be used exactly as output, or edited to add more details such as high, medium, low formatting and summary settings.

This manual should be read in conjunction with the MQEdit User Guide which contains detailed descriptions of MQEdit User Formats, in the chapter entitled “User Format Messages”.

This is the first version of this program and therefore contains limited functionality. If any users are interested in additional features, or the addition of languages other than COBOL, then we would be interested to hear from you.

1.2. Installation

Create a directory, say `makefmt`, and copy the installation zip file into it. You now need to unzip this file; you should be able to use any of the regular zip utilities to do it. For example, in Windows Explorer you should be able to right click on the file and select 'Extract All..'.

Once you have unzipped the file you should end up with a directory containing the following:

File	Description
`makefmt.exe`	The actual program itself
`makefmt.pdf`	This manual

1.3. Licensing

Unlike most other programs from MQGem Software, ***makefmt*** does not require a licence. It's sole purpose is to create user format files for use in other licensed products.

Chapter 2. Getting Started

In this chapter we'll show a simple example of how to use the *Make User Formats* (***makefmt***) program.

2.1. Running MakeFmt

In this example, we assume that you have a COBOL copybook on your local file system called `CUST.cpy`. It contains a definition of a COBOL structure that you want to convert to an MQEdit User Format.

Invoke MakeFmt with the following command:

```
makefmt -i CUST.cpy
```

This is the simplest example and will create an output file called `CUST.fmt` and the name of the top level structure inside that file will be `CUST`. If you prefer, the output filename and the name of the first structure can be specified on the command. See Chapter 4. MakeFmt Parameters on page 9 for more details.

In our example `CUST.cpy` contains the following short definition which demonstrates a number of features such as different field data types as well as variable arrays and packed decimal fields.

```
       01  CUST.
         10  FULL-NAME                PIC X(30).
         10  ADDRESS-LINE             PIC X(40) OCCURS 4 TIMES.
         10  CUSTOMER-NUMBER          PIC 9(5) COMP.
         10  NUM-INVOICES             PIC 9(5) COMP.
         10  INVOICE OCCURS 1 TO 10 TIMES DEPENDING ON NUM-INVOICES.
           20  INVOICE-DATE           PIC X(8).
           20  PO-NUMBER              PIC 9(10).
           20  INVOICE-TOTAL          PIC S9(5)V9(2) COMP-3.
         10  OVERALL-TOTAL            PIC S9(7)V9(2) COMP-3.
```

After we run the ***makefmt*** program, we have a file called `CUST.fmt` that has the following content.

```
struct INVOICE
{
  char           INVOICE_DATE[8];
  char           PO_NUMBER[10];
  packed(5.2)    INVOICE_TOTAL;
}

struct CUST
{
  char           FULL_NAME[30];
  char           ADDRESS_LINE[40][4];
  uint           CUSTOMER_NUMBER;
  uint           NUM_INVOICES;
  struct         INVOICE[NUM_INVOICES];
  packed(7.2)    OVERALL_TOTAL;
}
```

Now you have an MQEdit User Format file, you must associate it with the queue(s) and message format(s) that it should be used for. The MakeFmt program does not do this part for you.

Create a small file, say `mqedit.fmt` with the following contents.

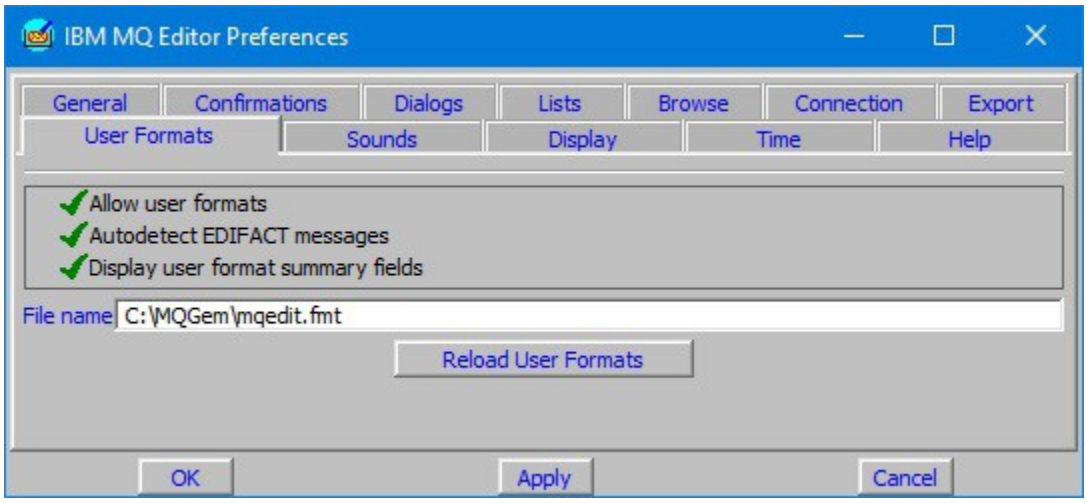
```
include "CUST.fmt"

queue CUSTOMER
{
  format $none
  {
    struct CUST;
  }
}
```

The first statement in this file includes the `CUST.fmt` file you just created by running ***makefmt***.

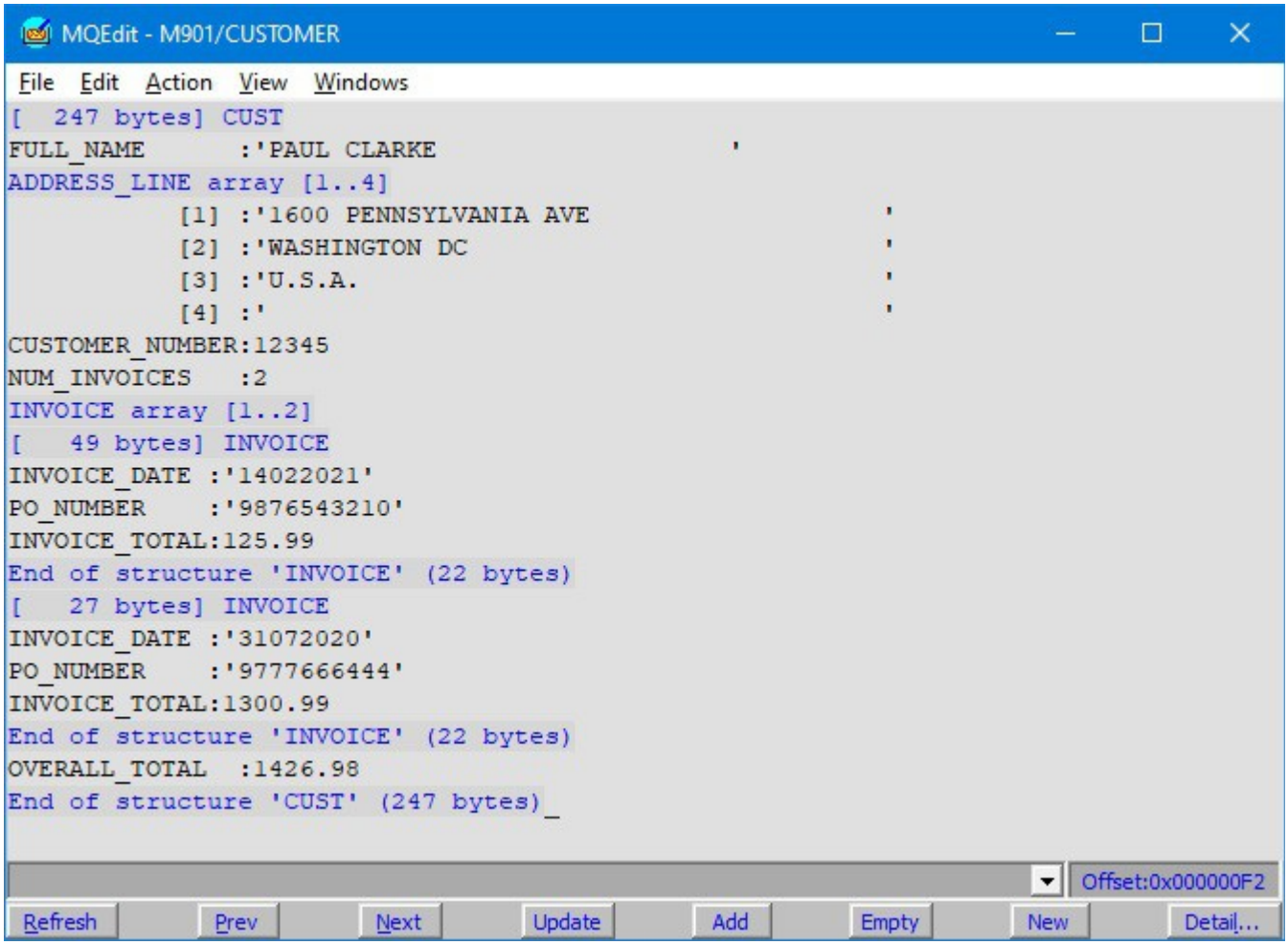
Next it states which queue(s), and then within that which message format(s), should use the `struct CUST` that is defined in the `CUST.fmt` file.

You may be wondering why we recommend making a separate file instead of adding this information to the file that ***makefmt*** just created. The reason for this is that we expect that you might have several different structures each created as an individual MQEdit User Format file from a copybook file, and that you would have a number of `include` statements. By keeping the generated files separate and including them in this way, you can re-run the generation step as often as required without having to change the queue and message format association file.

This small file, `mqedit.fmt`, is the one that you should tell the MQEdit program to use. Open the MQEdit preferences dialog, and go to the "User Formats" tab. Fill in the full path to your `mqedit.fmt` file, and press OK.



Now navigate to your message on your queue, to view your message displayed using your new User Format.



Chapter 3. Language Mapping (COBOL)

This chapter describes how ***makefmt*** maps a COBOL copybook into an MQEdit User Format file.

3.1. Data Types

Makefmt maps the following COBOL data types into MQEdit User format types as described in the table. If it is not mentioned, ***makefmt*** does not currently recognise the type. Please contact us to have this table extended to include data types that you need.

For more details on the MQEdit User Format Data Types, please refer to the MQEdit User Guide and the chapter entitled "User Format Messages".

COBOL Data Type	MQEdit User Format Data Type	Notes
`PIC X(n)`	`char[n]`	
`PIC A(n)`	`char[n]`	
`PIC 9(n)`	`char[n]`	
`PIC 9(n) COMP`	`uint8`	When n <= 2 (dependant on minimum binary size)
`PIC 9(n) COMP`	`ushort`	When n <= 4 (dependant on minimum binary size)
`PIC 9(n) COMP`	`uint`	When n between 5 and 9
`PIC 9(n) COMP`	`uint64`	When n > 9
`PIC S9(n) COMP`	`int8`	When n <= 2 (dependant on minimum binary size)
`PIC S9(n) COMP`	`short`	When n <= 4 (dependant on minimum binary size)
`PIC S9(n) COMP`	`int`	When n between 5 and 9
`PIC S9(n) COMP`	`int64`	When n > 9
`PIC 9(n) BINARY`		As for `PIC 9(n) COMP`
`PIC 9(n) COMP-4`		As for `PIC 9(n) COMP`
`PIC 9(n) COMP-5`		As for `PIC 9(n) COMP`
`COMP-1`	`float`	The distinction between IEEE floats and S/390 floats is made using the `Encoding` field of the message header. If the `Encoding` does not describe the correct floating point type, use an `override` clause in your MQEdit User Format file – see the MQEdit User Guide for more about this.
`COMP-2`	`double`	
`PIC S9(n) COMP-3`	`packed(n)`	
`PIC 9(n) COMP-3`	`upacked(n)`	
`PIC S9(n)V99 COMP-3`	`packed(n.2)`	
`PIC S9(n)V9(m) COMP-3`	`packed(n.m)`	
`PIC S9V99 COMP-3`	`packed(1.2)`	
`PIC 9(n) PACKED-DECIMAL`		As for `PIC 9(n) COMP-3`
`PIC S9(n)`	`opunch(n)`	If `TRAILING` or `LEADING` are specified these are translated into the `signright` or `signleft` options respectively.
`PIC S9(n) .. SEPARATE`	`digits(n)`	
`PIC S9(n)V99`	`opunch(n.2)`	
`PIC S9(n)V9(m)`	`opunch(n.m)`	
`PIC S9V99`	`opunch(1.2)`	

3.2. Groups

A COBOL group will become an MQEdit User Format `struct`. For example:-

```
         10  INVOICE.
           20  INVOICE-DATE           PIC X(8).
           20  PO-NUMBER              PIC 9(10).
           20  INVOICE-TOTAL          PIC S9(5)V9(2) COMP-3.
```

Would result in:-

```
struct INVOICE
{
  char           INVOICE_DATE[8];
  char           PO_NUMBER[10];
  packed(5.2)    INVOICE_TOTAL;
}
```

3.3. OCCURS

The `COBOL OCCURS` keyword results in an MQEdit User Format array. For example:-

```
         10  ADDRESS-LINE             PIC X(40) OCCURS 4 TIMES.
```

Would result in:-

```
  char           ADDRESS-LINE[40][4];
```

The `COBOL OCCURS … DEPENDING ON` keyword results in an MQEdit User Format variable array. For example:-

```
       01  CUST.
         10  NUM-INVOICES             PIC 9(5) COMP.
         10  INVOICE OCCURS 1 TO 10 TIMES DEPENDING ON NUM-INVOICES.
           20  INVOICE-DATE           PIC X(8).
           20  PO-NUMBER              PIC 9(10).
           20  INVOICE-TOTAL          PIC S9(5)V9(2) COMP-3.
```

Would result in:-

```
struct INVOICE
{
  char           INVOICE_DATE[8];
  char           PO_NUMBER[10];
  packed(5.2)    INVOICE_TOTAL;
}
struct CUST
{
  uint           NUM_INVOICES;
  struct         INVOICE[NUM_INVOICES];
}
```

Currently, MQEdit only supports the use of variable arrays when the length field is in the same `struct` as the field it is qualifying.

3.4. Other behaviours

There are various other decisions that ***makefmt*** takes when mapping a COBOL copybook into an MQEdit User Format file. These are covered in the following sections.

3.4.1. Minimum Requirements for storage on `COMPUTATIONAL/BINARY` fields

Some compilers have minimum requirements for the storage used for `COMPUTATIONAL` fields. For example, the smallest unit of storage may be 2 bytes, so even if you specify PIC 9 (only 1 digit), the compiler may reserve two bytes. Rather than guess what this value should be, ***makefmt*** requires you to tell it what this minimum requirement is **if** you have any 2 digit or fewer `COMP/BINARY` fields.

If you are unsure whether your copybook contains any of these, run ***makefmt*** without supplying this parameter, and if it is needed ***makefmt*** will report something like this:-

```
COMPUTATIONAL field 'NUM_INVOICES' with 1 digit found. Minimum Binary Size must be provided
using -b parameter
```

Then you can re-run ***makefmt*** supplying the appropriate value in the `-b` parameter. Read 4.2. Optional Parameters on page 9 for more details.

3.4.2. `REDEFINES`

By default, `COBOL REDEFINES` are ignored and the originally defined field name in the copybook is used for the field in the MQEdit User Format file.

To change this, provide the name or a wildcard mask to match the name using the `-r` parameter. Alternatively, use `-r*` (to say use all the redefines) and also use the `-R` parameter to provide the name or wildcard mask of those not to use. The order of these parameters is important, so please read 4.2.1. Combining included and excluded REDEFINED names on page 11 if you need to do this.

3.4.3. INDEXED BY

`COBOL INDEXED BY` are ignored.

3.4.4. Characters in field names

MQEdit User Format files do not allow arithmetic operators, e.g. plus (+), minus(-), divide (/) or multiply (*), in field names. COBOL does allow the dash or minus sign. These are sanitised by ***makefmt*** and replaced with an underscore (_).

3.4.5. SYNC

Use of the `SYNC` clause will result in MQEdit `align` statements being added to the MQEdit User Format if needed. Any unaligned numbers which are detected but do not have the `SYNC` clause, will not be aligned in the MQEdit User Format.

3.4.6. No name fields

All fields found with no name are named as `FILLER` in the MQEdit User Format file.

3.4.7. FILLER fields

There may be multiple fields with the 'name' `FILLER`. If so, these will be made unique by adding `_n`, where `n` is an appropriate number.

3.4.8. Level 88 VALUES

These will cause `enum` definitions to be created in the MQEdit User Format field and linked to the owning field name. Any level 88 lines with multiple values; or ranges of values (that is using the `THRU` or `THROUGH` keywords) are ignored when creating these `enum` definitions. Only single value definitions make sense.

For example, the following set of level 88 lines:-

```
       01  PERSON.
         05  NAME           PIC X(40).
         05  RELATIONSHIP   PIC X(01).
           88  VALID-RELATE VALUES 'S' 'P' 'F'.
           88  SPOUSE       VALUE  'S'.
           88  PARTNER      VALUE  'P'.
           88  FRIEND       VALUE  'F'.
```

Will result in the following `enum` definition. Note that the first 88 line defining the `VALID-RELATE` set of values is ignored.

```
enum RELATIONSHIP_enum
{
  "S", "SPOUSE";
  "P", "PARTNER";
  "F", "FRIEND";
}
struct PERSON
{
  char   NAME[40];
  char   RELATIONSHIP(RELATIONSHIP_enum);
}
```

By default `enum` definitions are created assuming that they contain the only values that will be seen in field. If this is not the case you can configure a field, which has an `enum`, to use the option `limval` to indicate to MQEdit that it should not be concerned when it sees a value it does not recognise. ***makefmt*** has an option to do this automatically if the majority of your value sets are defined this way, see `-Ol` in 4.2. Optional Parameters on page 9 for more details.

Chapter 4. MakeFmt Parameters

This chapter describes the various parameters to control how ***makefmt*** works.

4.1. Mandatory Parameters

- **-i** *<Input File Name>*
  This required parameter provides the name of the language file to be parsed. It must have a file extension from the following list to be interpreted as a particular programming language file.

Extension	Language
`.cpy`	COBOL
`.cbl`	COBOL
`.cob`	COBOL
`.ccp`	COBOL

4.2. Optional Parameters

- **-b** *<minimum binary size>*
  If your copybook contains any 1- or 2-digit fields, that is fields that could translate into a single byte integer (`int8` and `uint8`), you must supply this parameter to ensure ***makefmt*** knows what the minimum binary size for storage of such fields is for your COBOL compiler We list the defaults our research has indicated, but COBOL compilers may also have options to change their defaults. Consult your compiler reference for details.

  If your copybook contains no such fields, this parameter is not required. ***makefmt*** will tell you if it needs it.

Compiler	Minimum Binary Size
Enterprise COBOL for z/OS	2 bytes (see USAGE clause > Computational Items)
Micro Focus COBOL	1 byte (see Data Storage Options)

- **-c** *<min column, max column>*
  By default ***makefmt*** will only read code between columns 7 and 72, but if you need to change this, use this parameter.

- **-o <** *Output File Name***>**
  This is the name of the MQEdit User Format file to be created by ***makefmt***.

  If not provided, the file will be the structure name with the file extension `.fmt`

- **-O** *<Options>*
  There are various options that can be supplied with this flag. If required, more than one flag can be specified.
  For example `-Ofp`

Option	Description
`-Of`	Normally, ***makefmt*** will mark fields with a name of `FILLER`, or with a name containing the string '`FILLER`' as High detail. That is, MQEdit will only display these fields if you are looking at the message in High detail. For Medium and Low detail viewing these fields will not be shown. If you do not wish this to happen, use this flag to suppress the `high` keyword on these fields.
`-Ol`	Normaly, ***makefmt*** will create `enum` definitions assuming that all the possible values are known and part of the set. You can instruct ***makefmt*** to add the `limval` option to all fields with `enum` defintions using this option. Please see the MQEdit User Guide for more information on the `limval` option.
`-Oo`	Normally, ***makefmt*** will ask you to confirm that you want to over-write the output file. You can suppress that question with this option and allow ***makefmt*** to over-write the file without user confirmation. This can be handy when running ***makefmt*** in an unattended script.
`-Op`	Normally, ***makefmt*** will translate COBOL `COMP-3` or `PACKED-DECIMAL` fields into the MQEdit User Format `packed` data type (as described in 3.1 Data Types on page 5). If you are used to looking at packed decimal in raw hexadecimal form, you can use this option to have ***makefmt*** translate these fields into the MQEdit User Format `hex` data type.
`-Or`	If you use the `-r` and/or `-R` parameters, ***makefmt*** will report which `REDEFINED` fields are in use in place of the original field names. Use this option to suppress all these lines when you no longer need to be reminded of all of them.
`-Os`	Normally, makefmt will translate COBOL signed display fields into the MQEdit User Format opunch or digits data type (as described in 3.1 Data Types on page 5). If you are used to looking at these data types in raw hexadecimal form (as was the only option in previous releases of MQEdit, you can use this option to have makefmt translate these fields into the MQEdit User Format hex data type.

- **-p** *<prefix>*
  By convention, COBOL field names are often prefixed with the same characters as the overall copybook name. If you prefer to remove the first several characters from the field names to be shown in MQEdit, use this parameter to provide a prefix to be stripped from the beginning of field names.

- **-r** *<Included Redefine Name or Mask>*
  To use a redefined field name instead of the original field name, provide the name or a wildcard mask that matches it using this parameter. You can specify this parameter multiple times. If you want to use all of the redefined fields, specify `-r *`

  If some fields have multiple redefines, using `-r *` will select the first redefine unless you also use `-R` to exclude it. See 4.2.1. Combining included and excluded REDEFINED names below for how to use both together.

- **-R** *<Excluded Redefine Name or Mask>*
  To exclude the use of a redefined field name, specify the name or a wildcard mask that matches it using this parameter. You can specify this parameter multiple times. This is useful if you have included a number of redefined fields using `-r` and then you want to exclude some of them. See 4.2.1. Combining included and excluded REDEFINED names below for how to use both together.

  If your requirement is not to use any redefined fields, simply omit both the `-r` and `-R` parameters.

- **-s** *<Structure Name>*
  This is the name to be used for the very first structure created in the MQEdit User Format file.

  If not provided, the structure name will be created from the input file name with the path and file extension removed. For example, a file name of `include\CUST.cpy` will use a structure name of `CUST`.

- **-w** <*Warnings*>
  There are various warnings that can be enabled with this flag. If required, more than one flag can be specified.
  For example `-wra`

  If you want to switch on all warnings, just use `-w*`

Option	Description
`-wa`	Warn about unaligned number fields. While COBOL does not require number fields to be aligned on specific boundaries, if you are using this copybook for multi-language development, you might be interested in knowing which number fields are not aligned on appropriate boundaries.
`-wr`	Warn about redefined fields of differing lengths. While COBOL does not require redefined fields to be the same length as the field being redefined, many users prefer to insert padding or `FILLER` fields in their COBOL copybooks to ensure that they are.
`-w*`	All warning flags are on.

4.2.1. Combining included and excluded REDEFINED names

When you need to provide a wildcard pattern of `REDEFINED` field names to include and then within that pattern set some to exclude, you will need to use both the `-r` and `-R` parameters. MakeFmt will match against the set of `REDEFINED` names (or wildcard masks) in exactly the order you give them as parameters. Some common usage patterns follow.

To use all `REDEFINED` field names except those that match some particular pattern, provide the parameter with the most specific pattern first. So, for example:-

```
makefmt -i CUST.cpy -R *DATE* -r*
```

To use two overlapping patterns, one to include and one to exclude, provide the most specific pattern first. So, for example, to ensure the name `ABC` is used, but that the names `ABD` and `ABE` are not, use these parameters:-

```
makefmt -i CUST.cpy -r ABC* -R AB*
```

If you're at all unsure, pay attention to the output messages that MakeFmt writes to the screen showing which redefined field names were used in place of original field names, e.g.

```
REDEFINE definition 'BIRTHDATE-FULLFMT' used instead of 'BIRHTDATE-PARTS'
```

If you have a lot of these lines and no longer need to see them all, this reporting can be suppressed with the `-Or` flag.

Chapter 5. Trouble Shooting

5.1. Service

As with any software product of any complexity it always possible that there are bugs in the code. If you notice strange behaviour or the program crashes then by all means raise a bug report by sending an email to support@mqgem.com. Before you do ensure you are using the latest version of the software. The latest version is always available for download here https://www.mqgem.com/makefmt_download.html.

If upgrading doesn't help and you are certain it is a product bug then please do send an email to support@mqgem.com Please be as specific as possible about the problem you are having and how the problem can be recreated. Provide the command used to invoke ***makefmt***, the language file being processed, and what you expected the output to be. Remember that the more detail you put in to your problem description then the faster your problem can be resolved.

End of document

MQGem Software Limited

www.mqgem.com