

ecCodes BUFR decoding

Fortran 90 and Python API – part 1

Marijana Crepulja

Introduction:

- Fortran 90 subroutines to decode BUFR data
- Python subroutines to decode BUFR data
- Practical examples



Fortran API: Decoding

Fortran API: Decoding

- Open a file according to a mode

call `codes_open_file (file, filename, mode, status)`

file: the opened file to be used in all file functions
filename: name of the file to be opened
mode: open mode can be 'r' (read) or 'w' (write)
status: `CODES_SUCCESS` if OK, integer value on error

Input arguments

Output arguments

- Close a file

call `codes_close_file (file, status)`

file: the file to be closed
status: `CODES_SUCCESS` if OK, integer value on error

Fortran API: Error handling

- In case of error, if the `status` parameter (optional) is not given, the program will exit with an error message.
- The error message can be gathered with `codes_get_error_string`.

```
if (status /= 0) then
  call codes_get_error_string (status, err_msg)
  print*, 'ecCodes Error: ', trim(err_msg), ' error=', status
  stop
end if
```

- Error codes are listed under:
http://download.ecmwf.int/test-data/eccodes/html/group_errors.html



Fortran API: Decoding

Input arguments

Output arguments

- Load in memory a message from a file

```
call codes_bufr_new_from_file (file, ibufr, status)
```

file: the file opened with `codes_open_file`

ibufr: id of the message loaded in memory

status: `CODES_SUCCESS` if OK, `CODES_END_OF_FILE` at the end of file, or error code

The message can be accessed through its message ID and it will be available until `codes_release`.

- Release a BUFR message

```
call codes_release(ibufr, status)
```

ibufr: the id of the message to be released

status: `CODES_SUCCESS` if OK or error code

Fortran API: Decoding

Input arguments

Output arguments

- Get the value of the key in the message

call `codes_get (ibufr, key, value, status)`

`ibufr`: id of the message loaded in memory

`key`: key of a variable

`value`: values of variable/array

`status`: `CODES_SUCCESS` if OK, or error code



`value` can be a integer(4), real(4), real(8) or a character.

Type of the returned values depend on the variable declaration.

`integer (kind=4) :: integer_values`

`real (kind=8) :: real_values`

`character (len=string_size) :: string_values`

Fortran API: Decoding

Input arguments

Output arguments

- This function supports the **allocatable** array attribute.

call codes_get (ibufr, key, value, status)

Type of the returned value depend on the array declaration.

```
integer (kind=4),      dimension(:), allocatable :: integer_array  
real     (kind=8),      dimension(:), allocatable :: real_array
```



if(allocated(array)) deallocate(array)

Fortran API: Decoding

- Get values of string array

call `codes_get_string_array (ibufnr, key, value, status)`

`ibufnr`: id of the message loaded in memory

`key`: key of a variable

`value`: values of variable/array

`status`: `CODES_SUCCESS` if OK, or error code



Type of the returned `value` depends on the array declaration.

`character (len=string_size), dimension(:), allocatable :: string_array`

! The array of strings must be allocated before it is passed to `codes_get_string_array`

`allocate (string_array)`

! Remember to deallocate

`if(allocated(string_array)) deallocate(string_array)`

Input arguments

Output arguments

Fortran API: Decoding

- Getting number of subsets in the message

```
call codes_get(ibufr,'numberOfSubsets',numberOfSubsets)
```

Input arguments
Output arguments

- Getting master table version number

```
call codes_get(ibufr,'masterTablesVersionNumber',masterTablesVersionNumber)
```

- Getting values of expanded descriptors

```
call codes_get(ibufr, 'expandedDescriptors', value, status)
```

- Getting originating Centre

```
call codes_get(ibufr, 'bufrHeaderCentre', value, status)
```

Fortran API: Decoding

- We need to instruct ecCodes to unpack the data section
call `codes_set(ibufr, 'unpack', 1)`

Input arguments
Output arguments



Note: If you do not set 'unpack' and want to read the data section, ecCodes returns
ECCODES ERROR : get: latitude Key/value not found



Fortran API: Missing values

- Each element in the data section of a BUFR can be missing.
- ecCodes provides a simple way for the user to check if the value of an element is missing by comparing with two constants:

`CODES_MISSING_LONG` for integer values

`CODES_MISSING_DOUBLE` for real values

The constants are available in Python, Fortran 90 and C and the user needs to compare with the appropriate constant depending on the type of the variable used.



Fortran API: Missing values

- Declare value as double precision

```
real(kind=8), dimension(:), allocatable :: value
```

- Get the values

```
call codes_get(ibufr, 'key', value)

do i=1,size(value)
  ! compare with double precision missing
  if (value(i)/=CODES_MISSING_DOUBLE) then
    ! process non missing values
  else
    ! process missing values
  endif
end do
```

Fortran API: Decoding example

```
call codes_open_file(file, 'filename', 'r')
call codes_bufnr_new_from_file(file, ibufnr, status)
do while (status/=CODES_END_OF_FILE)
    ! Read the BUFR headers info
    call codes_get(ibufnr, 'numberOfSubsets', numberOfSubsets)
    ! unpack the data values
    call codes_set(ibufnr, "unpack", 1)
    ! Read the data section
    call codes_get(ibufnr, 'latitude', lat)
    ! Release the message
    call codes_release(ibufnr)
    ! Load the next message
    call codes_bufnr_new_from_file(file, ibufnr, status)
end do

call codes_close_file(file)
```

Input arguments
Output arguments



Python API: Decoding



Python API: Decoding

Input arguments
Output arguments

- Open the input BUFR file

```
file = open(filename, 'rb')
```

file: the opened file to be used in all file functions

filename: name of the file to be opened



- Close the BUFR file

```
file.close()
```

file: the file to be closed





Python API: Decoding

Input arguments
Output arguments

- Load in memory a message from a file

```
ibufr = codes_bufc_new_from_file(file)
```

file: the file opened with `codes_open_file`

ibufr: id of the message loaded in memory



The message can be accessed through its message ID in all ecCodes function calls and it will be available until `codes_release`

- Free the memory for the message referred to by `ibufr`

```
codes_release(ibufr)
```



Python API: Exception handling

All ecCodes functions throw the following exception on error: **CodesInternalError**
This is the base class for all other exceptions. The full list is here:

<https://confluence.ecmwf.int/display/ECC/Python+exception+classes>

try:

```
example(output_filename)
```

except CodesInternalError as err:

```
sys.stderr.write(err.msg + '\n')
```





Python API: Decoding

- Get the value of a scalar from a message

```
value = codes_get(ibufr, key)
```

ibufr: id of the message loaded in memory

key: key of the variable

value: value of variable

Input arguments

Output arguments

- Get the array values of a variable from a message

```
value = codes_get_array(ibufr, key)
```

ibufr: id of the message loaded in memory

key : key of the variable

value: an array as a NumPy array.



Type of the value can be integer, float or str.



Python API: Decoding

- Getting the number of subsets in the message

```
numberOfSubsets = codes_get(ibufr, 'numberOfSubsets')
```

Input arguments

Output arguments

- Getting the master table version number

```
masterTablesVersionNumber = codes_get(ibufr, 'masterTablesVersionNumber')
```

- Getting values of expanded descriptors

```
expandedDescriptors = codes_get(ibufr, 'expandedDescriptors')
```

- Getting originating Centre

```
centre = codes_get(ibufr, 'bufrHeaderCentre')
```

Python API: Decoding

Input arguments
Output arguments

- We need to instruct ecCodes to unpack the data section

```
codes_set(ibufr, 'unpack', 1)
```



Note: If you not set 'unpack' and want to read data section ecCodes returns

ECCODES ERROR : get: latitude Key/value not found



Python API: Decoding example

```
file = open(filename, 'rb')

# loop over the messages in the file
while 1:

    ibufr = codes_bufnr_new_from_file(file)
    if ibufr is None:
        break

    # Read the BUFR headers info
    numberOfSubsets = codes_get (ibufr, 'numberOfSubsets')
    # unpack the data values
    codes_set(ibufr, 'unpack', 1)

    # Read the data values
    airTemperature = codes_get(ibufr, 'airTemperature')

    codes_release(ibufr)
file.close()
```

Input arguments
Output arguments



Set up environment

- F90 interface
use `eccodes`
- Python interface
`from eccodes import *`
- At ECMWF environment variables `ECCODES_INCLUDE` and `ECCODES_LIB` are used for compiling C and Fortran programs.



Compile and run the program

- Fortran 90 interface

```
gfortran -o myprogram myprogram.f90 \  
    $ECCODES_INCLUDE $ECCODES_LIB  
./myprogram
```

- Python interface

```
python myprogram.py
```



Practical

- Navigate to your \$SCRATCH

```
cd $SCRATCH
```

- Copy the material for the practical

```
cp -r ~trx/ecCodes/2019/bufr_api_decode ./
```

- There are subdirectories for F90 and python

```
cd F90
```

```
cd python
```

- The directories are named by practical number

```
e.g. cd bufr_decode_practical1
```

- Have a look at the README

- Have fun



Practical 1: Decode SYNOP data

1. Open the **synop.bufr** file in read mode
2. Load message
3. Loop over messages
4. Decode and print:
 - unexpandedDescriptors
 - expandedDescriptors
5. '**unpack**' the data section
6. Decode and print:
 - latitude
 - longitude
 - airTemperature
7. Release the message
8. Close the BUFR file



codes_open_file

codes_bufr_new_from_file

codes_set (ibufr,'unpack',1)

codes_get

codes_release

codes_close_file

What happens if you do not set 'unpack'?

Accessing value by rank

Input arguments

Output arguments

Fortran API

- Reading value of variable by rank

```
call codes_get(ibufr, '#2#key', value, status)
```

Python API

- Reading value of variable by rank

```
height = codes_get (ibufr, '#2#height' )
```

- Reading array values of variable by rank

```
latitude = codes_get_array (ibufr, '#2#latitude' )
```

Practical 2: Decode TEMP data

1. Open **temp.bufr** in read mode
2. Load message
3. Loop over messages
4. '**unpack**' the data section
5. Get values for:
 - 'shipOrMobileLandStationIdentifier'
 - 'latitude',
 - 'longitude'
 - 'height'
6. Using accessing variables by rank to decode the second **windSpeed** and **windDirection**
7. Release the message
8. Close the BUFR file

Helpful Tips

```
codes_open_file
codes_bufr_new_from_file
codes_set (ibufr,'unpack',1)
codes_get
codes_release
codes_close_file
```

Practical 3: Decode SYNOP data

1. Open the **synop_2messages.bufr** in read mode
2. Load the messages in memory
3. Loop over messages
4. **'unpack'** the data section
5. Decode and print:
- **'stationOrSiteName'**
6. Release the message
7. Close the BUFR file



Reminder

In Fortran API

```
character (len=string_size), dimension(:), allocatable :: stationOrSiteName
```

! The array of strings must be allocated before is passed to codes_get_string_array

```
call codes_get_size(ibufr,'stationOrSiteName',n)
```

```
allocate (stationOrSiteName(n))
```



Helpful
Tips

```
codes_open_file
```

```
codes_bufr_new_from_file
```

```
codes_set (ibufr,'unpack',1)
```

```
codes_get
```

```
codes_release
```

```
codes_close_file
```

Practical 4: Decode SYNOP data

1. Open the **synop_with_confidence.bufr** in read mode
2. Load the messages in memory
3. Loop over messages
4. **'unpack'** the data section
5. Decode and print:
 - **'nonCoordinatePressure'**
 - its **'percentConfidence'**
6. Release the message
7. Close the BUFR file



How to access confidence??

Use the key 'nonCoordinatePressure->percentConfidence'

Helpful
Tips

```
codes_open_file  
  
codes_bufr_new_from_file  
codes_set (ibufr,'unpack',1)  
  
codes_get  
  
codes_release  
  
codes_close_file
```

References

- ecCodes:

<https://software.ecmwf.int/wiki/display/ECC/ecCodes+Home>

- BUFR tables:

<https://software.ecmwf.int/wiki/display/ECC/BUFR+tables>

- Error codes are listed under:

http://download.ecmwf.int/test-data/eccodes/html/group__errors.html

