Sub-hourly support in MARS

Manuel Fuentes

Requirement gathering

- Need description of the system(s) producing sub-hourly data
  - Do we need to mix sub-hourly with medium-range, eg
  - 15’, 30’, 1hour, 90’, 2hour, 24 hour, 240 hours
- Support for non-instantaneous fields, eg, accumulations, maximum, minimum, ..
  - Precipitation from 15’ to 75’
  - Maximum 2t from 90’ to 3hours

Current handling of time-step

- Step, mostly instantaneous

  step = 12/24
- Step ranges, daily means, weekly-means

  step = 0-24/24-48/96-264

- MARS Client:
  – long
  – long-long

grib_api

```
> grib_ls -pdataDate,stepRange,dataType,shortName data.fc
dataDate    stepRange   dataType    shortName
20160305    24          fc          2t
20160305    23-24       fc          mn2t
20160305    24          fc          tp
> grib_ls -pmars.date,mars.time,mars.step,mars.param,mars.type data.fc
mars.date   mars.time   mars.step   mars.param  mars.type
20160305    1200        24          167.128     fc
20160305    1200        24          202.128     fc
20160305    1200        24          228.128     fc


> grib_ls -pdataDate,stepRange,dataType,shortName data.taem
data.taem
dataDate    stepRange   dataType    shortName
20160303    0-168       taem        2t
20160303    96-264      taem        2t
```

MARS Server: Step

```
struct MarsStep: public RootMarsType {
        typedef double value_type;
        typedef double persistent_value_type;
        static std::string specName() { return "MarsStep"; }
        static const char*  name() { return "step"; }
        static void getValues(const MarsRequest&
     r,std::vector<value_type>& v)
                { r.getValues(name(),v); }
 static value_type valueFromFile(NodeHook* h,eckit::Ordinal i)
                { value_type v; ASSERT(h);
     dynamic_cast<ArchiveGribHook&>(*h).getValue(i,name(),v);
     ASSERT(long(v) == v); return v; }
        static void retrievePatch(const MarsRequest&,
     std::vector<value_type>&,
                                                             const
     PVector<persistent_value_type>&) {}
};
```

MARS Server: StepRange

```
struct MarsStepRange: public RootMarsType {
    typedef StepRange value_type;
    typedef StepRange persistent_value_type;
    static std::string specName() { return
  "MarsStepRange"; }
    static const char*  name() { return "step"; }
    static void getValues(const MarsRequest&
  r,std::vector<value_type>& v);
    static value_type valueFromFile(NodeHook*
  h,eckit::Ordinal i);
    static void retrievePatch(const MarsRequest&,
  std::vector<value_type>&,const
  PVector<persistent_value_type>&);
};
```

What are your suggestions ?

Possible solutions

- New keyword *mstep*

  mstep=5/15/30/.../1440/14400

  mstep=5/15/30/.../24h/240h

- New keyword *stepUnits*

  stepUnits=minutes

  step=5/15/30/.../1440/14400

- Unit in step

  step=15m/30m/45m/1/75m/90m

Implications

- grib_api/ecCodes
  - – What would *grib_ls –m* return ?
- MARS Client
  - – Not many, probably handle internally in minutes or seconds
  - – Existing requests MUST work
- MARS Server
  - – Create new C++ class:
    - MarsSubHourlyStep
    - MarsStepSeconds