ANNEX D – Dataset Loading Algorithm (Dataset Selection)

Preconditions

An inventory for each ~~*item* (data~~**Data**~~-~~ ~~coverage~~**Coverage**~~)~~ contains:

- A geo polygon describing the **Data Coverage**~~data-coverage~~: *geo = polygon(~~item~~dataCoverage)*;
- A set of scale ~~bands~~band indices: ~~*scaleBands*~~ *S_{SBI} = ScaleBands(~~item~~dataCoverage)*;
- An associated dataset: *dataset(~~item~~dataCoverage)*.

A projection *pro* that can:

- Convert ~~a~~ geographic ~~-~~polygons *geo* to device~~-~~ polygons: *poly = pro(geo)*;
- Convert device ~~-~~polygons *poly* to geographic~~-~~ polygons: *geo = ~pro(poly)*.

D-1 Scale Bands

A list~~s~~ of scale bands will be used for the algorithm. Each scale band is defined by the denominator of its minimum and maximum scales and will be accessed by an index.

~~I~~iIndex	~~Minimum Scale~~minimumScale	~~Maximum Scale~~maximumScale	Remarks
1	∞~~NULL~~	10,000,000	For all scale smaller than 1:10,000,000
2	10,000,000	3,500,000	
3	3,500,000	1,500,000	
4	1,500,000	700,000	
5	700,000	350,000	
6	350,000	180,000	
7	180,000	90,000	
8	90,000	45,000	
9	45,000	22,000	
10	22,000	12,000	
11	12,000	8,000	
12	8,000	4,000	
13	4,000	3,000	
14	3,000	2,000	
15	2,000	1,000	

The following algorithm associates a scale band with the denominator of a scale~~ band~~:

```
Algorithm GetScaleBand($scale_d$)
Input: The denominator of ~~A~~a scale
Output The index of ~~the ~~a scale band
  1. If $scale_d$ ~~scale <~~≥ ScaleBands[1].maximumScale~~[1]~~
       a. Return 1
  2. For index = 2 ~~->~~ through 15
       a. If (ScaleBands[index].minimumScale ≥ $scale_d$)~~[index] <= scale A~~AND ($scale_d$ ~~scale~~ ≥~~<~~
          ScaleBands[index].maximumScale~~[index]~~)
            i. Return index
  3. Return 15
```

The set of scale bands for a~~n item~~ **Data Coverage** ~~"data-coverage" with its *minScale* and *maxScale* would~~is ~~be~~ defined as:

```
Algorithm ~~scaleBands~~ScaleBands(~~item~~dataCoverage)
Input: ~~Item as~~ a Data Coverage~~data-coverage~~
Output: A set of associated scale band indices $S_{SBI}$
  ~~1. minDS – The minimum display scale of the coverage maxDS – The maximum display scale
     of the coverage~~
  ~~2.~~1. Create an empty set $S_{SBI}$~~S~~
  2. minimumScale = dataCoverage.minimumDisplayScale
       a. If minimumScale = null
            i minimumScale = ∞
  3. If ~~minDS~~ minimumScale >~~<~~ ScaleBands[1].maximumScale~~[1]~~
       a. $S_{SBI}$~~S~~ = $S_{SBI}$~~S~~ ∪ 1
  4. For index = 2 through~~->~~ 15
       a. If ~~If~~ m~~ax~~in(dataCoverage.minimumDisplayScale,
          ScaleBands[index].minimumScale~~[index]~~) ~~<~~≥
          max~~in~~(dataCoverage.maximumDisplayScale,
          ScaleBands[index].maximumScale~~[index]~~)
            i $S_{SBI}$~~i. S~~ = $S_{SBI}$~~S~~ ∪ index
  5. Return $S_{SBI}$~~S~~
```

D-2 Dataset Coverage Selection Process

The next algorithm shows the selection process of the data coverages.

The idea is to find all data coverages for the scale band that contains the scale parameter and select those which overlap the viewport. The viewport ~~will be~~should then be modified ~~in a way~~so that it only defines the part that is ~~still not~~yet to be covered.

If this part is not empty the algorithm will proceed with the next smaller scale band until the remaining viewport is empty or there is no smaller ~~more~~ scale band to investigate.

```
Algorithm SelectDataCoverages(INV, $scale_d$, viewport, pro)
Input: An inventory of Data Coverages INV
    The denominator of a scale~~A~~ $scale_d$~~scale~~ ~~for which the data coverages will be selected~~
    (usually the denominator of the display scale)
    A device-polygon viewport describing the device area that should be covered with data
    A projection pro
Output: A set of Data Coverages $S_{DC}$~~inventory items S~~
  1. Create an empty set $S_{DC}$~~$S = \emptyset$~~
  2. index~~SB~~ $= GetScaleBand($$scale_d$~~scale~~$)$
  3. While $viewport \neq \emptyset$ do
     a. For each~~all~~ dataCoverage~~item~~ in INV
        i. If index ~~SB~~ $\in$ ~~scaleBands~~ScaleBands(dataCoverage~~item~~) AND~~A~~
           $(pro(poly($dataCoverage~~item~~$)) \cap viewport) \neq \emptyset$
           1. $S_{DC}$~~S~~ $=$ $S_{DC}$~~S~~ $\cup$ dataCoverage~~item~~
           2. $viewport = viewport \setminus pro(poly($dataCoverage~~item~~$))$
     b. index ~~SB~~ $=$ index ~~SB~~ $- 1$
     c. If index ~~SB~~ $= 0$
        i. Return $S_{DC}$~~S~~
  4. Return $S_{DC}$~~S~~
```

Comments:

Row	Description
1.	Create an empty set of **Data Coverages**~~inventory items~~
2.	Get the index of the scale band ~~to which~~associated with *$scale_d$*~~*scale*~~ ~~belong~~ and assign it to the variable ~~*SB*~~ *index*
3.	~~As long as~~While the *viewport* area is not empty
3.a	Loop over all **Data Coverages**~~items~~ in the inventory
3.a.i	If *index* ~~*SB*~~ is an element of the scale bands ~~of~~ associated with the **Data Coverage**~~item~~ and the projected coverage polygon of the **Data Coverage**~~item~~ overlaps the *viewport*
3.a.i.1.	Add the **Data Coverage**~~item~~ to *S_{DC}*~~*S*~~
3.a.i.2.	Remove the projected coverage polygon from the *viewport*, The *viewport* will now only define the uncovered part of the original *viewport*
3.b.	Decrement *index* ~~*SB*~~
3.c.	If *index* ~~*SB*~~ equals to zero ~~(No scale band left to investigate)~~
3.c.i.	Return the collected result (there were no scale bands left to investigate)
4.	Return the collected result (the viewport was filled by the **Data Coverages** in S_{DC})

D-3 Dataset Selection

The final algorithm selects the set of datasets associated with the **Data Coverages** selected by the previous algorithm. Each selected dataset should be loaded in its entirety.

```
Algorithm SelectDatasets($S_{DC}$)
Input: A set of Data Coverages $S_{DC}$
Output: A set of datasets $S_{DS}$, each of which should be loaded in its entirety.
    1. Create an empty set $S_{DS}$
    2. For each dataCoverage in $S_{DC}$ do
        a. $S_{DS} = S_{DS} \cup dataset(dataCoverage)$
    3. Return $S_{DS}$
```