Preconditions

An inventory for that each *item* contains

- A geo polygon describing the coverage: *polygon(item)*
- A set of scale bands: *scaleBands(item)*
- An associated dataset: *dataset(item)*

A projection *pro* that can

- Convert a geo-polygon *geo* to device-polygon: *pro(geo)*
- Convert device-polygons *poly* to geo-polygons: *~pro(poly)*

Scale bands

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

Index	Min Scale	Max Scale	Remarks
1	NULL	1:10,000,000	For all scale smaller than 1:10,000,000
2	1:10,000,000	1: 3,500,000	
3	1:3,500,000	1:1,500,000	
4	1:1,500,000	1:700,000	
5	1:700,00	1:350,000	
6	1:350,000	1:180,000	
7	1:180,000	1:90,000	
8	1:90,000	1:45,000	
9	1:45,000	1:22,000	
10	1:22,000	1:12,000	
11	1:12,000	1:8,000	
12	1:8,000	1:4,000	
13	1:4,000	1:3,000	
14	1:3,000	1:2,000	
15	1:2,000	1:1,000	

The following algorithm associate a scale with a scale band:

Algorithm GetScaleBand(scale)
Input A scale
Output The index of the scale band

1. **If** *scale < maxScale[1]*
   a. **Return** 1
2. **For** *index* = 2 -> 15
   a. **If** $minScale[index]\ >=\ scale\ \wedge scale\ <\ maxScale[index]$
      i. Return *index*
3. **Return** 15

The set of scale bands for a coverage with minScale and maxScale would be defined as:

Algorithm *GetScaleBandsForCoverage(minDS, maxDS)*
Input: *minDS* – The minimum display scale of the coverage
maxDS – The maximum display scale of the coverage
Output: A set of associated scale band indices *S*

1. Create an empty set *S*
2. **If** $minDS < maxScale[1]$
   a. $S = S \cup 1$
3. **For** index = 2 -> 15
   a. If $max(minDS, minScale[index]) < min(maxDS, maxScale[index])$
      i. $S = S \cup index$
4. **Return** S

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

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

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

Algorithm *SelectDataSets*(*INV, scale, viewport, pro*)
Input: A inventory *INV*
A *scale* for that the datasets will be selected (usually the display scale)
A device-polygon *viewport* describing the device area that should be covered with data
A projection *pro*
Output: A set of inventory items *S*

1. $S = \emptyset$
2. $SB = GetScaleBand(scale)$
3. **While** $viewport \neq \emptyset$ **do**
   a. **For** all *item* in *INV*
      i. **If** $SB \in scaleBands(item) \wedge (pro(poly(item)) \cap viewport) \neq \emptyset$
         1. $S = S \cup item$
         2. $viewport = viewport \setminus pro(poly(item))$
   b. $SB = SB - 1$
   c. If $SB = 0$
      i. **Return** *S*
4. **Return** *S*

Comments:

Row	**Description**
1.	Create an empty set of inventory items
2.	Get the scale band to which *scale* belong and assign it to the variable *SB*
3.	As long as *the* viewport area is not empty
3.a.	Loop over all items in the inventory
3.a.i.	If SB is an element of the scale bands of the item and the projected coverage polygon of the item overlaps the viewport
3.a.i.1.	Add the item to S
3.a.i.2.	Remove the coverage polygon from the viewport, The viewport will now only define the uncovered part of the original viewport.
3.b.	Decrement SB
3.c.	If SB equals to zero (No scale band left to investigate)
3.c.i.	Return the collected result
4.	Return the collected result