**Draft appendix for S-98 Annex C Data loading algorithms.**

**Notes.**

1. Taken from v1.3.0 of the S-101 Product Specification
2. Agreed by S-101PT 13 that it will migrate into S-98 Annex C
3. Comments required – also input on new algorithm proposed at S-101PT13
4. For inclusion in S-98 Annex C 1.3.0 onwards.
5. Appears to be out of date in respect of MaximumDisplayScale / optimumDisplayScale ?
6. Should any parts of 4.6 and 4.7 be included somewhere, particularly the highlighted section in red in 4.6 (which was discussed at PT13
7. Existing numbering has been preserved.

Introduction is constructed from parts of the existing text and references to parts of S-98 Annex C where parts already exist.

**Introduction**.

Algorithms for dataset loading and unloading; and rendering (display) within a navigation system are prescribed for S-101 in order for the appropriate ENC to be viewed at the mariner’s selected viewing scale.

This will simplify the process for navigation systems, giving clear and concise rules on how and when data is loaded and unloaded; and the order at which datasets are to be displayed.

The concept of navigation purpose is restricted for use in presenting ENCs in a visual catalogue and must not be used for determining which dataset should be displayed, see clause XX-XX.

Details of the dataset loading and data display algorithms are contained within this Appendix.

Note 1: These algorithms only address loading and display related to visualization within the system graphics window. The application may need to load other datasets to satisfy requirements related to alerts processing, such as MSC.530(106) A11.2.

Note 2: Light sectors [and other features which may span dataset boundaries]. It should be possible, on request, for the mariner to be capable of identifying the colour of the sectors affecting the ship, even if the lights involved are off the display, and, in general provision must be made to ensure that all relevant features are portrayed.

## Overview – Display scale ranges

Scales and scale ranges are defined in Clause XXX-XXX.

A scale range of a dataset is used to indicate a range of scales between which a producer considers the data is intended for use. (See clause 4.7 for how datasets are to be loaded and unloaded within a navigation system.) The smallest scale is defined by the **minimum display scale** and the largest scale by the **optimum display scale**. The **maximum display scale** indicates the scale that the Data Producer considers that the “grossly overscaled” warning should be triggered. These scales must be set at one of the scales specified in clause 3 (spatial resolutions).

There must not be overlapping scale ranges (that is, overlaps between values of **optimum display scale** and **minimum display scale**) between datasets covering the same geographical area. ECDIS behaviour in the event of an overlap are dealt with in clause XX-XX.

*When the mariner’s selected viewing scale (MSVS) is smaller than the value indicated by* ***minimum display scale****, features within the* ***Data Coverage*** *feature are not displayed, except where the System Database does not contain a dataset covering the area at a smaller scale, in which case the dataset will be continuously displayed.*

When the MSVS is larger than the value indicated by **optimum display scale**, the overscale indication, in the form of an overscale factor covering the area that is overscale, must be shown. When at own ship’s position a dataset with a larger **optimum display scale** than the MSVS is available, an indication is required and must be shown on the same screen as the chart display. This is descrbied in Clause XX-XX

When the MSVS is larger than the value indicated by **maximum display scale**, the overscale indication, in the form of an overscale factor and, additionally, a pattern covering the area that is overscale, must be shown to indicate that the data is “grossly overscaled”.

Within ENC schemes it is preferable that the scale ranges for different datasets covering the same geographical area to be continuous (see clause 4.5.3). However, where the scale ranges are non-continuous, the ECDIS will display the larger scale dataset until the MSVS is equal to or at smaller scale than the **optimum display scale** of the next smaller scale dataset This is described in Cause XX-XX**.**

# ANNEX D – Dataset Loading Algorithm (Dataset Selection) and Dataset Display Order (Dataset Rendering)

**Preconditions**

An inventory for each **Data Coverage** contains:

* A geo polygon describing the **Data Coverage**: *polygon(dataCoverage)*;
* A set of scale bands: *scaleBands(dataCoverage)*;
* An associated dataset: *dataset(dataCoverage)*.

A projection *projection* that can:

* Convert geographic polygons *geoPolygon* to device polygons: *projection(geoPolygon)*;
* Convert device polygons *polygon* to geographic polygons: *~projection(polygon)*.

1. **Scale Bands**

A lists of scale bands will be used for the algorithm. Each scale band is defined by its minimum and maximum scale denominators and will be accessed by an index. Note that the table below contains the denominators of the scale; for example. 22,000 is the denominator value associated with the scale 1:22,000. Whenever scales are compared in these algorithms the numerical comparison is based on scales not on scale denominators.

|  |  |  |  |
| --- | --- | --- | --- |
| ***index*** | ***minimumScale*** | ***maximumScale*** *(maximum)* | **Remarks** |
| 1 | NULL (∞) | 10,000,000 | For all values larger than 10,000,000 |
| 2 | 10,000,000 | 10,000,000 < *maximum* ≤ 3,500,000 |  |
| 3 | 3,500,000 | 3,500,000 < *maximum* ≤ 1,500,000 |  |
| 4 | 1,500,000 | 1,500,000 < *maximum* ≤ 700,000 |  |
| 5 | 700,000 | 700,000 < *maximum* ≤ 350,000 |  |
| 6 | 350,000 | 350,000 < *maximum* ≤ 180,000 |  |
| 7 | 180,000 | 180,000 < *maximum* ≤ 90,000 |  |
| 8 | 90,000 | 90,000 < *maximum* ≤ 45,000 |  |
| 9 | 45,000 | 45,000 < *maximum* ≤ 22,000 |  |
| 10 | 22,000 | 22,000 < *maximum* ≤ 12,000 |  |
| 11 | 12,000 | 12,000 < *maximum* ≤ 8,000 |  |
| 12 | 8,000 | 8,000 < *maximum* ≤ 4,000 |  |
| 13 | 4,000 | 4,000 < *maximum* ≤ 3,000 |  |
| 14 | 3,000 | 3,000 < *maximum* ≤ 2,000 |  |
| 15 | 2,000 | 2,000 < *maximum* ≤ 1,000 |  |

The following algorithm associates a scale denominator with a scale band:

**Algorithm** *GetScaleBand(scale)*

**Input**: A scale

**Output** The index of the scale band

1. **If** *scale < maximumScale[1]* 
   1. **Return** 1
2. **For** *index* = 2 to 15
   1. **If** 
      1. **Return** *index*
3. **Return** 15

The set of scale bands for a **Data Coverage** with its *minimumDisplayScale* and *maximumDisplayScale* is defined as:

**Algorithm** *scaleBands(dataCoverage)*

**Input**: A **Data Coverage**

**Output:** A set of associated scale band indices *S*

1. *minimumDisplayScale* – The minimum display scale of the coverage (if not defined it is assumed that the scale is 1:∞ -> 0)  
   *maximumDisplayScale* – The maximum display scale of the coverage
2. Create an empty set *S*
3. **If**
4. **For** index = 2 to 15
   1. If
5. **Return** S
6. **Dataset Coverage Selection Process**

The next algorithm shows the selection process of the **Data Coverage** features.

The idea is to find all **Data Coverage** features for the scale band that contains the scale parameter and select those which overlap the viewport. The viewport should then be modified in a way that it only defines the part that is 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 scale band to investigate.

|  |
| --- |
| **Algorithm** *SelectDataCoverages*(*inventory, scale, viewport, projection*)  **Input**: A inventory of **Data Coverage** features *inventory*  A *scale* for which the **Data Coverage** features will be selected (usually the display scale)  A device-polygon *viewport* describing the device area that should be covered with data  A projection *projection*  **Output**: A set of **Data Coverage** features *S*   1. 𝑆 = ∅ 2. *ScaleBand* = 𝐺𝑒𝑡𝑆𝑐𝑎𝑙𝑒𝐵𝑎𝑛𝑑(𝑠𝑐𝑎𝑙𝑒) 3. **While** 𝑣𝑖𝑒𝑤𝑝𝑜𝑟𝑡 ≠ ∅ **do**    1. **For** all *dataCoverage* in *inventory*       1. **If** *ScaleBand* ∈ 𝑠𝑐𝑎𝑙𝑒𝐵𝑎𝑛𝑑𝑠(*dataCoverage*) AND (*𝑝𝑟𝑜jection*(*𝑝𝑜𝑙𝑦gon*(*dataCoverage*)) ∩ 𝑣𝑖𝑒𝑤𝑝𝑜𝑟𝑡) ≠ Ø          1. 𝑆 = 𝑆 ∪ *dataCoverage*          2. 𝑣𝑖𝑒𝑤𝑝𝑜𝑟𝑡 = 𝑣𝑖𝑒𝑤𝑝𝑜𝑟𝑡 \ *𝑝𝑟𝑜jection*(𝑝𝑜𝑙*ygon*(*dataCoverage*))    2. *ScaleBand* = *ScaleBand* – 1    3. **If** *ScaleBand* = 0       1. **Return** *S* 4. **Return** *S* |

Comments:

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

Note that the algorithm above selects **Data Coverage** features. The system will then load the associated datasets. In the case where multiple selected **Data Coverage** features are associated with the same dataset, this dataset will be loaded only once.

1. **Data Display Algorithm**
   1. **Data display algorithm based on drawing index**
      1. **General**

After the data-coverages are selected and the associated datasets are loaded the chart display will be generated by:

1. Create a set of drawing instructions for each dataset. This step is called portrayal and defined by the rules in the Portrayal Catalogue.
2. Render the drawing instructions as described below.

Notes:

* Datasets can only be portrayed entirely, there is no mechanism to only portray single data coverages.
* The algorithm assumes that the rendering is made by using a kind of the ‘Painters algorithm’. This means an opaque fill will completely obscure what has been rendered at this position before. This does not mean that any implementation must follow this approach; other techniques like Z-Buffer technique may be used. The algorithm will not give implementation details, any implementor has the freedom to reach the desired result in the most effective way.
  + 1. **The Rendering Algorithm**

The first step is to group the datasets into subsets which we will denote ‘Layers’. The criteria for the separation will be the value for the attributes **drawing index** and **minimum display scale** for the **Data Coverage** feature(s) for the dataset. Note that all data coverages within a dataset must have the same values for **minimum display scale** and **drawing index**, and data sets with the same minimum display scale or the same drawing index are not allowed to overlap. To be precise, the union of all data coverages of one dataset must not overlap the union of the data coverages of another dataset with the same minimum display scale or drawing index.

1. Datasets which share a common (non-null) drawing index are grouped together in single layers.
   1. The minimum display scale of these layers is the smallest minimum display scale (the largest scale denominator) of the component datasets.
2. From the remaining datasets, those which share a common minimum display scale are grouped together in single layers.
3. Layers from A and B which share a common minimum display scale are grouped together in single layers.

The ‘Layers’ are then sorted by their minimum display scale and sequentially rendered starting with the smallest minimum display scale.

**Algorithm**: *RenderChartImage*

**Input**: A set of datasets *dataSets*

A drawing device

1. Split the set dataSets into sub-sets denoted *layer0*, *layer1*, … *layern* such that the drawing index of each dataset in one *layerx* is not null but is otherwise the same.
   1. Assign a minimum display scale to each layer from the smallest minimum display scale (the largest scale denominator) of the component datasets.
2. Split the remaining dataSets (those where drawing index is null) into sub-sets denoted *layern+1*, *layern+2*, … such that the minimum display scale of each dataset in one *layerx* is the same.
3. Combine layers which share a common minimum display scale
4. Sort the *layer1 .. layern* by its associated minimum display scale
5. Clear the drawing device (e.g. by filling the drawing device with the NODTA colour or pattern.
6. Iterate over all *layerx* starting with the smallest minimum display scale
7. Render the layer with the algorithm *RenderLayer*

NOTE 1: For the sake of simplicity the concept of display planes (that is, under and over radar) is not considered here. Without loss of generality the algorithm can be used multiple times to create the images for each display plane. One way of achieving it is to split the output of the portrayal into subsets; one for each display plane and run the algorithm for each subset. However, the painters algorithm cannot be used to render data in the over radar display plane since there will not be Skin of the Earth objects present to obscure underlying layers.

NOTE 2: The algorithm as described here does not distinguish between official and non-official data. It could be achieved by taking this into account during the grouping of the input datasets.

* + 1. **The Algorithm RenderLayer**

This algorithm describes how the datasets of one layer; that is, those that have the same minimum display scales and/or drawing indices are rendered.

**Algorithm**: RenderLayer

**Input**: A set of datasets *dataSets* that have the same minimum display scale and/or drawing indices  
 A drawing device

1. **For** each display priority *displayPriority* starting with the smallest
   1. Collect the active drawing instructions from each dataset’s display instructions that are assigned to *displayPriority*
   2. *Note: while null instructions should not be rendered, they should show in the pick report below the other instructions from that collection*
   3. Render the area instructions from that collection
   4. Render the line instructions from that collection
   5. Render the point instructions from that collection
   6. Render the text instructions from that collection

NOTES:

1a: Rendering must take the *viewingGroup(s)*, *scaleMinimum*, *scaleMaximum, date dependency, and any other* properties of the display instruction which may affect the instructions visibility into account. (See S-100 Part 9)

1f: When rendering text, an implementation may take into account the guidance in S-100 Part 9 regarding text rendering to adjust this algorithm as needed to enhance the readability of text.

## Dataset loading and display order

Figures 4-7 to 4-9 below are intended to assist in understanding how the datasets should be displayed in the system graphics window:

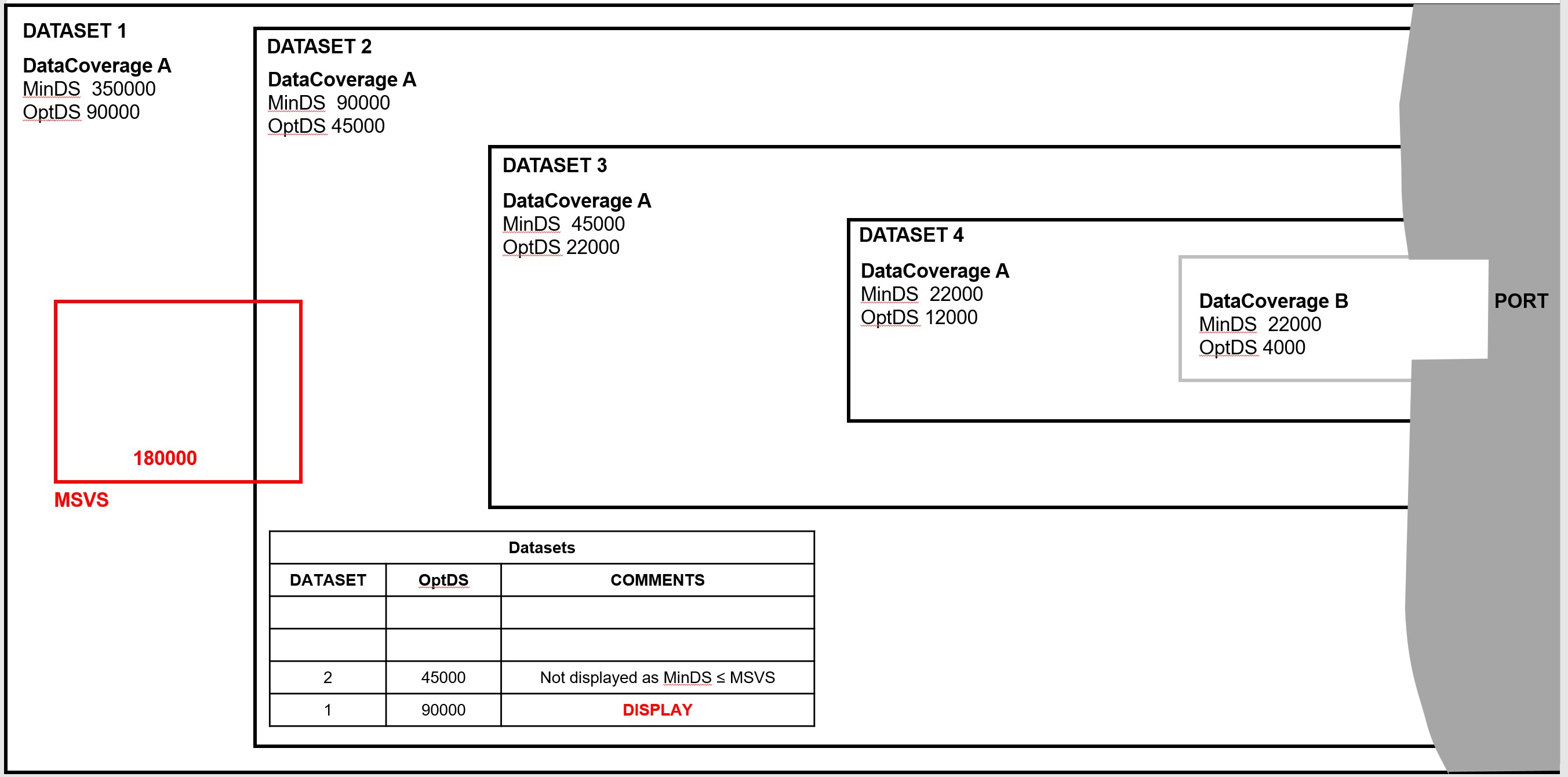


Figure 4-7 – Dataset loading – scenario 1

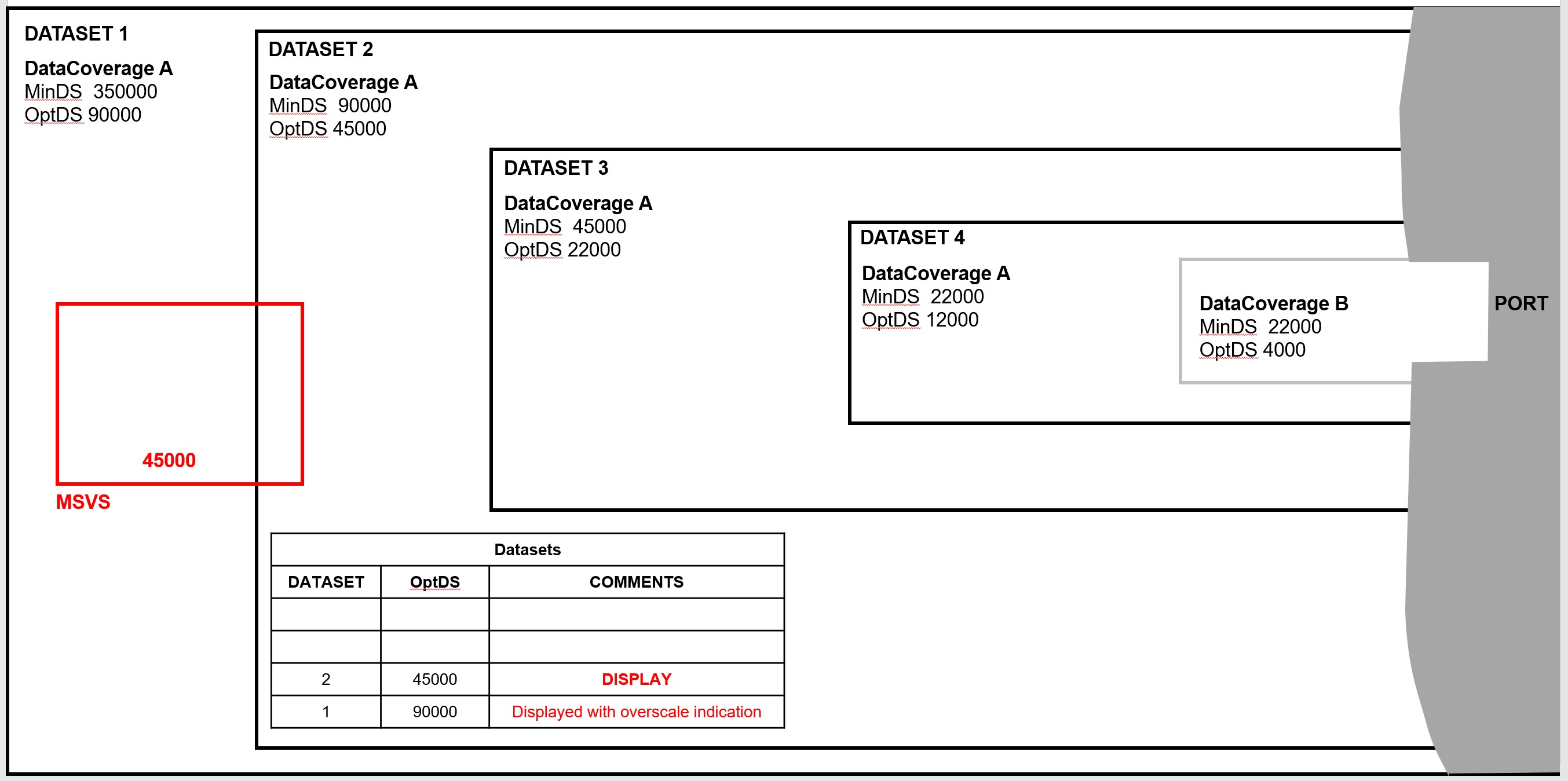
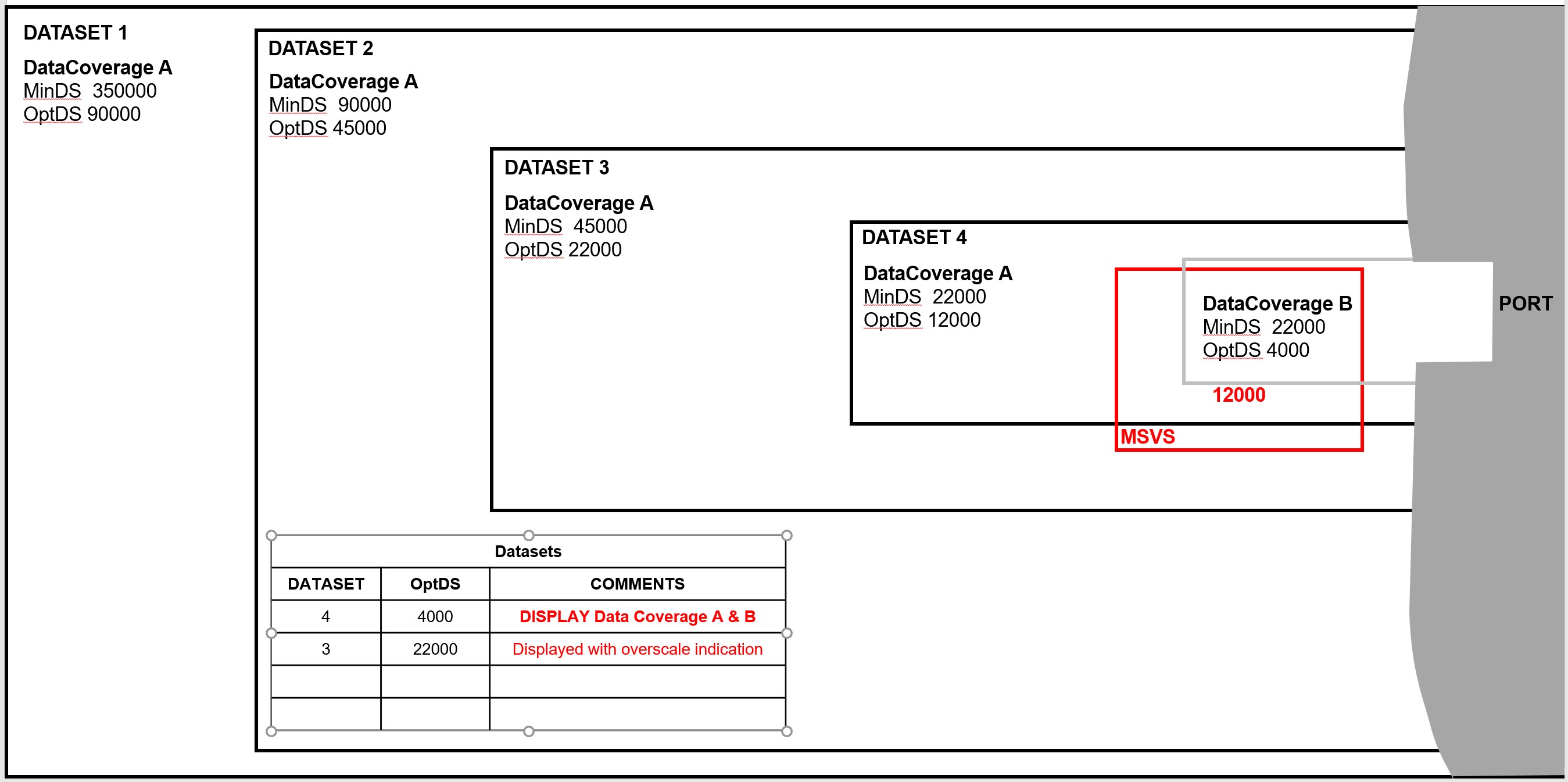


Figure 4-8 – Dataset loading – scenario 2



**Figure 4-9 – Dataset loading – scenario 3**