MapInfo Virtual Raster User Guide

Sam Roberts June 2020

Introduction

This document describes the MapInfo Virtual Raster (MVR) format as it was defined in MapInfo Pro version 2019 Patch 2. It provides example XML that you can use to author your own MVR files and describes the XML schema for MVR files.

A virtual raster is a description of a raster that can be loaded, displayed and operated upon in MapInfo Pro just like any other raster format. The raster data is acquired or generated on demand, at run time. The MapInfo Virtual Raster (MVR) is a simple XML file - using the ".MVR" file extension or you can use an ".xml" extension if you prefer - that can be loaded into MapInfo Pro via the raster handler interface.

Understanding MVR

A virtual raster is structured just like a normal raster which means it needs to have fields and bands defined, cell size and tile origin defined and all the other properties of a raster clearly defined. In the XML, this is done by creating a RasterInfo element.

However, most of the raster properties can be derived from the raster sources that are referred to by the virtual raster. Consequently, the user usually has to do little other than link raster sources to fields and bands.

Virtual raster data is cached by the raster engine just like any other raster data. If the requested data is not in the cache, then the virtual raster engine will be invoked to provide the data. This will trigger a process that may be deep and complex, depending on how that virtual raster data is defined.

Ultimately, data will be acquired from some source raster. The MVR will require this data to be interpolated into its own geometry which can change depending on what resolution level of the MVR you are requesting data for. This will trigger a requirement to acquire source data at some resolution level which may be anywhere from underview levels through to overview levels. For this reason, it is important that source rasters are in MRR format or have overview cache files.

Displaying an MVR in MapInfo Pro

To display an MVR in MapInfo Pro, load it as a Raster via the usual Open Table interface.

Creating an MVR in MapInfo Pro Advanced (2019.2 or later)

There are two user interface tools in MapInfo Pro Advanced that will create an MVR for you.

The first is the "Virtual Raster" operation. This tool allows you to select raster source files and combine them together into a multi-banded MVR.

The second is the "Warp Image" operation. This tool has an option to output an MVR which will perform the image warping on the fly by invoking a warp operation as a raster source. You can

display the MVR immediately and, if you are happy with the warp transform, go on to execute the processing operation (or just work with the MVR thereafter).

Both of these operations are highly specific and produce only certain kinds of MVR files. To produce MVR files with other capabilities you need to author the XML yourself.

Creating an MVR

To create an MVR you will need a text editor or an XML editor. I use “Notepad++” which is free and very capable. A good place to start is to copy and paste some example XML from this document, or from other MVR files you have worked on, and develop the XML from that starting point.

Structure of an MVR

Within the VirtualRaster root element you will define a set of global objects that you will refer to by name elsewhere in the MVR. Global objects may also refer to other global objects by name.

You will also define a RasterInfo element that, at the very least, will declare how the fields or bands of the MVR acquire data from globally define sources. It may also declare other structural information about the MVR.

Data for fields and bands has to come from somewhere and these are defined as sources. There are three kinds of sources – Raster, Operation and Render Algorithm. The one thing that all raster sources need is a unique name. The objects will be referred to by this name throughout the XML.

Raster Sources

A raster source is another raster. It can be in any supported raster format. In almost all circumstances it will be a requirement that the raster has an overview pyramid cache and it is generally a good idea that full base level statistics have been computed for the raster and stored in a GHX file. If the raster is an MRR then these things are all stored internally.

When you declare a raster source you just supply the filename of the raster and any driver preference information that will be used to modify the driver behaviour when it opens the raster. You do not supply things like field and band indices – these will be specified when you refer to the raster source elsewhere in the MVR. So an MVR can access any of the fields and bands in a raster source.

A raster source can contain more than one raster file. If so, then it is an expectation of the system that all the rasters will have the same structure. For example, you could have a collection of located aerial images in a directory. You could gather all of these images into a raster source using a filename wildcard.

Operation Sources

An operation source is an operation list that contains a processing operation. Even though you can store multiple operations in an operation list, you should only use one in a MVR. The processing operation will generally draw upon other raster sources and the output of the operation will be a

new raster. The operation may define the structure of this raster. The MVR will then ingest that raster data into its own fields and bands.

Render Algorithm Sources

Rendering algorithms are used to acquire rendered output from the "new" rendering engine. This rendering engine is quite distinct from the mechanism that usually renders rasters in MapInfo Pro which invokes raster handlers.

You can write rendering algorithms as ".MRD" XML files and load them into MapInfo Pro via the normal raster Open Table interface. This is the usual way that the new rendering engine would be invoked. MVR provides another way to access the rendering engine. A rendering algorithm will output an RGBA pixel buffer which can be exposed as an Image field in an MVR.

If you have multiple rendering algorithms embedded in an MVR it makes it easy to swap between rendering algorithms by simply changing the field on the raster toolbar.

RasterInfo

Every MVR requires a RasterInfo element that defines all the properties of the raster. This can include things like the cell size but in practice it usually just includes definitions for all the fields and/or bands and what sources those fields or bands are connected to. From this data, the raster engine is able to deduce what the raster structure is and assign appropriate properties.

Links to Community Articles

There are a number of articles in the Community archive which provide additional resources on the use of virtual rasters and rendering algorithms.

Working with MapInfo Virtual Rasters

Virtual Raster Deep Dive – Part 1 – What is a Virtual Raster?

Virtual Raster Deep Dive – Part 2 – Satellite Multispectral Imagery

Raster Rendering Deep Dive – Part 1 – Default Algorithms

Raster Rendering Deep Dive – Part 2 – LUTcolor Layers

Raster Rendering Deep Dive – Part 3 – RGBColor Layers

Raster Rendering Deep Dive – Part 4 – Image Layers

Raster Rendering Deep Dive – Part 5 – Fixed Data-Color Transforms

Polygonising an Image Raster using Virtual Rasters

Warping Capability in MapInfo Pro

Denmark DTM – Building a trillion cell terrain raster

Satellite Multispectral Data Preparation

Examples

1. Minimum.mvr

The first example, shown below, demonstrates how to combine multiple individual raster source files together into a multi-banded MVR.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <!-- Raster source declarations -->
    <Raster Name="Blue">
        <Filename>Landsat8/LC08_L1TP_089083_20191106_20191115_01_T1_B2.TIF</Filename>
        <DriverPreferences ValidateOrUpdateStatistics="True" VehicleTIFFNullValue="0"/>
    </Raster>
    <Raster Name="Green">
        <Filename>Landsat8/LC08_L1TP_089083_20191106_20191115_01_T1_B3.TIF</Filename>
        <DriverPreferences ValidateOrUpdateStatistics="True" VehicleTIFFNullValue="0"/>
    </Raster>
    <Raster Name="Red">
        <Filename>Landsat8/LC08_L1TP_089083_20191106_20191115_01_T1_B4.TIF</Filename>
        <DriverPreferences ValidateOrUpdateStatistics="True" VehicleTIFFNullValue="0"/>
    </Raster>
    <!-- Raster Info structure -->
    <RasterInfo>
        <FieldInfo Name="Landsat 8">
            <BandInfo Name="Band 2 - Blue">
                <Raster Name="Blue" Field="0" Band="0" PrimaryRaster="True"/>
            </BandInfo>
            <BandInfo Name="Band 3 - Green">
                <Raster Name="Green" Field="0" Band="0"/>
            </BandInfo>
            <BandInfo Name="Band 4 - Red">
                <Raster Name="Red" Field="0" Band="0"/>
            </BandInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

Three raster sources are declared and uniquely named, each of which refers to a TIFF file in the Landsat8 directory. The MVR structure is then defined with a single field and three bands. A raster source is linked, by name, to each band and the field index and band index required from the TIFF is declared. The first raster source is declared as the primary raster.

Note the format of comments (the green text) in the XML. This syntax in XML is quite unforgiving and, in general, I advise against using comments.

In this example I have declared driver preferences for all raster sources which improve the data quality. I ensure that statistics for these source rasters are computed and stored and I define the null value which, in the case of Collection 1 Landsat data, is not internally defined.

2. LC08_L1TP_089083_20191106_20191115_01_T1_DNData.mvr

The second example is a more complex example of combining multiple files into a multi-banded MVR. This MVR has additional driver preference flags to ensure that the Landsat 8 QA band is loaded correctly and the overview pyramid is constructed appropriately. Also, interpolation settings for the QA band are restricted to nearest neighbour. In this MVR, there are source rasters that have different cell sizes. I choose to declare the Panchromatic band, which has the smallest cell size, as the primary raster. This has an impact when you use this raster in a processing operation as the operation will be performed at this cell size.

3. SeamlessTable.mvr

This example, shown below, simulates a seamless table. The raster source acquires all TIFF files in the “Seamless” directory and consumes these in an Image field.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="Seamless">
        <Filename>Seamless\*.TIF</Filename>
        <DriverPreferences MountAsRasterFieldType="Image" GlobalCompressQualitySpace="True"/>
    </Raster>
    <RasterInfo>
        <FieldInfo Name="UK Topography">
            <Type>Image</Type>
            <BandInfo Name="Band 1 - Topo">
                <Raster Name="Seamless" Field="0" Band="0"/>
            </BandInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

The TIFF files use a color palette but I use a driver preference to force these to be loaded as RGB images, which is more visually pleasing. Also, I use a driver preference to change the image compression codec used in the overviews – biasing it towards quality over quantity. Because this data is topographic imagery, I want high image quality and lossless compression. Note that setting these flags will not force overview cache files to be regenerated if they were not generated using these flags initially. Sometimes you need to delete old PPRC files to ensure that new ones are created using the properties you desire.

I explicitly declare the field of type Image and link the color band to the raster source.

4. SeamlessTableDC.mvr

This builds on the previous example. The topographic imagery has a background blue which I would like to remove. The background color is RGB(213,244,248) and I remove this using a data conditioning object. The RGB value is declared and all cells with this color value are returned as invalid. I also define the interpolation settings to nearest neighbour to make sure that this color remains pure at all levels of zoom.

This is not the only way to remove this color from the raster, nor is it the best. Another option is to set the null value in the TIFF driver to this color using a driver preference. See "SeamlessTableNull.mvr" as an example of doing this. This example uses a different set of TIFF files because the overview pyramid cache will be different – the null color will have made no contribution to this pyramid.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="Seamless">
        <Filename>Seamless\*.TIF</Filename>
        <DriverPreferences MountAsRasterFieldType="Image" GlobalCompressQualitySpace="True"/>
    </Raster>
    <DataConditioning Name="RemoveColor">
        <Value>16315605</Value>
    </DataConditioning>
    <RasterInfo>
        <FieldInfo Name="UK Topography">
            <Type>Image</Type>
            <BandInfo Name="Band 1 - Topo">
                <Raster Name="Seamless">
                    <Field>0</Field>
                    <Band>0</Band>
                    <DataConditioningName>RemoveColor</DataConditioningName>
                    <UnderviewInterpolation>Nearest</UnderviewInterpolation>
                    <InterpolationNearest>True</InterpolationNearest>
                </Raster>
            </BandInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

5. PanTRNaturalColor.mvr

This example illustrates how to use a rendering algorithm in a virtual raster and to consume it as an Image field.

```
<VirtualRaster Version="1.0">
    <Raster Name="RasterSource" Filename="LC08_L1TP_089083_20191106_20191115_01_T1_PanTRRGB.mvr"/>
    <DataTransform Type="Sigmoid" Name="DataTransform">
        <RangeType>Percentile</RangeType>
        <RangeMin>0.005</RangeMin>
        <RangeMax>0.995</RangeMax>
        <SigmoidK>-0.25</SigmoidK>
    </DataTransform>
    <RenderAlgorithm Version="1.0" Name="4_3_2">
        <Layer Type="RGBColor" Name="Natural Color (4_3_2)">
            <Component Type="Red" Enable="True">
                <Raster Name="RasterSource" Field="0" Band="0"/>
                <DataTransformName>DataTransform</DataTransformName>
            </Component>
            <Component Type="Green" Enable="True">
                <Raster Name="RasterSource" Field="0" Band="1"/>
                <DataTransformName>DataTransform</DataTransformName>
            </Component>
            <Component Type="Blue" Enable="True">
                <Raster Name="RasterSource" Field="0" Band="2"/>
                <DataTransformName>DataTransform</DataTransformName>
            </Component>
            <Component Type="Intensity" Enable="False"/>
            <Component Type="Opacity" Enable="False"/>
        </Layer>
    </RenderAlgorithm>
    <RasterInfo Name="LC08_L1TP_089083_20191106_20191115_01_T1 Pan-sharpened TOA Reflectance Images">
        <BaseTileSize>256,256</BaseTileSize>
        <UnderviewTileSize>256,256</UnderviewTileSize>
        <OverviewTileSize>256,256</OverviewTileSize>
        <FieldInfo Type="Image" Name="Natural Color (4_3_2)">
            <BandInfo Name="Natural Color (4_3_2)">
                <RenderAlg Name="4_3_2"/>
            </BandInfo>
        </FieldInfo>
        <FieldInfo Type="Image" Name="Natural Color (4_3_2)">
            <RenderAlg Name="4_3_2"/>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

This example renders pan-sharpened, top of atmosphere corrected Landsat 8 data. The rendering algorithm uses an RGBColor layer to implement a Natural Color rendering. A number of other similar examples are included but they include multiple rendering algorithms,

each of which they consume in a different field of the MVR. The difference between them is the source of the raster data (as described below). Note that the pan-sharpened imagery uses a chain of intermediate virtual rasters to apply this operation in real time.

LC08_L1TP_089083_20191106_20191115_01_T1_DNImages.mvr – uses digital number data directly from the original TIFF files.

LC08_L1TP_089083_20191106_20191115_01_T1_PanImages.mvr – pan-sharpened digital number data.

LC08_L1TP_089083_20191106_20191115_01_T1_PanTRImages.mvr – sources data from virtual rasters that use calculator operations to apply a “top of atmosphere” correction to the digital number data.

LC08_L1TP_089083_20191106_20191115_01_T1_TRImages.mvr – pan-sharpened TOA data.

In the example, you will see that the rendering algorithm is consumed in two different fields. This is simply to illustrate how to consume an algorithm directly in a field and also how to consume it in a band. Also, note the specification of Base, Overview and Underview tile sizes in the raster. In this raster, I use a small tile size to improve multi-threaded rendering performance by reducing the number of thread collisions at tiles being held for write access.

6. TOATemp.mvr

This example introduces the use of operations in MVR bands and the calculator operation.

```
<VirtualRaster Version="1.0">
    <Raster Name="RasterSource9">
        <Filename>Landsat8/LC08_L1TP_089083_20191106_20191115_01_T1_B10.TIF</Filename>
        <DriverPreferences ValidateOrUpdateStatistics="True" VehicleTIFFNullValue="0"/>
    </Raster>
    <Raster Name="RasterSource10">
        <Filename>Landsat8/LC08_L1TP_089083_20191106_20191115_01_T1_B11.TIF</Filename>
        <DriverPreferences ValidateOrUpdateStatistics="True" VehicleTIFFNullValue="0"/>
    </Raster>
    <RasterOperationList Name="TOATemp9">
        <Calculator Name="TOATemp9">
            <Variable Name="DN">
                <Raster Name="RasterSource9" Field="0" Band="0"/>
            </Variable>
            <Expression>1321.0789 / ln(1 + (774.8853/(3.3420E-04 * DN + (0.10000))))</Expression>
        </Calculator>
    </RasterOperationList>
    <RasterOperationList Name="TOATemp10">
        <Calculator Name="TOATemp10">
            <Variable Name="DN">
                <Raster Name="RasterSource10" Field="0" Band="0"/>
            </Variable>
            <Expression>1201.1442 / ln(1 + (480.8883/(3.3420E-04 * DN + (0.10000))))</Expression>
        </Calculator>
    </RasterOperationList>
    <RasterInfo Name="LC08_L1TP_089083_20191106_20191115_01_T1 Top Of Atmosphere Spectral Data">
        <BaseTileSize>256,256</BaseTileSize>
        <UnderviewTileSize>256,256</UnderviewTileSize>
        <OverviewTileSize>256,256</OverviewTileSize>
        <FieldInfo Name="TOA Brightness Temperature">
            <BandInfo Name="Band 10 - TIRS1">
                <DataType>REAL4</DataType>
                <Operation Name="TOATemp9"/>
            </BandInfo>
            <BandInfo Name="Band 11 - TIRS2">
                <DataType>REAL4</DataType>
                <Operation Name="TOATemp10"/>
            </BandInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

The MVR draws upon two Landsat 8 thermal bands and applies a correction to convert the thermal digital numbers to “top of atmosphere” brightness temperature. This is a standard correction that can be applied to Landsat data and it is based on metadata constants that ship with each scene.

Two operations lists are defined, each of which contains a calculator operation. The calculator operations declare a variable and link this to a raster source and field and band. This variable is then used in the calculator expression. Finally, we consume each operation list in separate bands. In these bands we promote the data type to REAL4 (a 32 bit floating point type) as the input data type is unsigned integer.

Additional similar examples are provided.

LC08_L1TP_089083_20191106_20191115_01_T1_TOAData.mvr – TOA computations for radiance, reflectance and brightness temperature.

LC08_L1TP_089083_20191106_20191115_01_T1_DNIndex.mvr – Calculator operations are used to compute simple band ratios and formulas for Landsat 8 digital numbers.

LC08_L1TP_089083_20191106_20191115_01_T1_TRIndex.mvr - Calculator operations are used to compute simple band ratios and formulas for Landsat 8 TOA corrected data. In this case a MVR processing chain is used.

LC08_L1TP_089083_20191106_20191115_01_T1_Pan.mvr – Calculator expressions are used to perform pan-sharpening on different Landsat 8 band combinations. In this case multiple variables are declared and more complex expressions are used.

LC08_L1TP_089083_20191106_20191115_01_T1_PanTR.mvr - Calculator expressions are used to perform pan-sharpening on different Landsat 8 band combinations. In this case multiple variables are declared and more complex expressions are used and the input bands are derived from another MVR.

LC08_L1TP_089083_20191106_20191115_01_T1_PanRGB.mvr – This MVR uses the pan-sharpened RGB values and splits them back out into individual Red, Green and Blue components. These are then gathered into separate fields which are suitable for rendering via an RGBColor layer.

LC08_L1TP_089083_20191106_20191115_01_T1_PanTRRGB.mvr - This MVR uses the pan-sharpened TOA corrected RGB values and splits them back out into individual Red, Green and Blue components. These are then gathered into separate fields which are suitable for rendering via an RGBColor layer.

LC08_L1TP_089083_20191106_20191115_01_T1_QAValues.mvr – Calculator expressions are used to pick individual bit masks out of the QA band in Landsat 8 data and to expose these as individual bands of a virtual raster.

7. Pure.mvr

This example illustrates how to create a raster from a pure calculator expression or mathematical formula, without any other inputs.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <RasterOperationList Name="Formula">
        <Calculator Name="Formula">
            <Expression>100*cos(degtorad(sqrt(pow(GetX(),2)+pow(GetY(),2))))</Expression>
        </Calculator>
    </RasterOperationList>

    <RasterInfo Name="ComputedRaster">
        <CoordinateSystem>CoordSys NonEarth Units "m"</CoordinateSystem>
        <BaseTileSize>1024,1024</BaseTileSize>
        <GridSize>2048,2048</GridSize>
        <FieldInfo Name="Computed">
            <Type>Continuous</Type>
            <Compression>LZ4,0</Compression>
            <OverviewCellCoverage>Coverage_Half</OverviewCellCoverage>
            <ValidFlagPerBand>True</ValidFlagPerBand>
            <Transform Type="Simple">
                <Origin>0,0</Origin>
                <CellSize>1,1</CellSize>
            </Transform>
            <BandInfo Name="Computed surface">
                <DataType>REAL4</DataType>
                <StoreDataType>REAL4</StoreDataType>
                <RestrictDecimals>True</RestrictDecimals>
                <MaxDecimals>4</MaxDecimals>
                <Clip>False</Clip>
                <Transform>False</Transform>
                <PredEncode>False</PredEncode>
                <NullValueType>Mask</NullValueType>
                <BandValueType>Standard</BandValueType>
                <Operation Name="Formula"/>
            </BandInfo>
            <EventInfo EditType="Total">
                <Time>February-13-2018 08:23:11</Time>
            </EventInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

As there are no source rasters from which to acquire structural information, you need to define the RasterInfo structure manually. Critically, you need to define the GridSize (columns and rows) and you need to define a Transform for each field in which you define the origin coordinate and the cell size. You need to link each band to a calculator operation in which you declare a pure formulaic expression which does not require any raster input variables. I have defined an event as every field must have at least one event – but if you do not define one the system will create one for you.

There are some issues with rasters of this kind, because virtual rasters do not honour the grid size. If you make sure the tile size is a integer multiple of the grid size, then the base level will be correctly sized. However, this does not apply to overview and underview levels. The best way to use one of these rasters is to immediately convert them to a real raster.

The example shown contains information that is not strictly required. For an example that uses only the minimum amount of XML, see “PureMinimum.mvr”.

8. Event.mvr

This example illustrates how to build an MVR that contains multiple events, each of which is linked back to an individual raster. You will need a bespoke viewer application (like TestWindow) to visualise this example as the ability to view individual events is not implemented

in MapInfo pro Advanced 2019 patch 2. The MVR declares 10 raster source files and consumes each of these in an event that is declared within a field.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="Raster0" Filename="Events/Event0.mrr"/>
    <Raster Name="Raster1" Filename="Events/Event1.mrr"/>
    <Raster Name="Raster2" Filename="Events/Event2.mrr"/>
    <Raster Name="Raster3" Filename="Events/Event3.mrr"/>
    <Raster Name="Raster4" Filename="Events/Event4.mrr"/>
    <Raster Name="Raster5" Filename="Events/Event5.mrr"/>
    <Raster Name="Raster6" Filename="Events/Event6.mrr"/>
    <Raster Name="Raster7" Filename="Events/Event7.mrr"/>
    <Raster Name="Raster8" Filename="Events/Event8.mrr"/>
    <Raster Name="Raster9" Filename="Events/Event9.mrr"/>
    <RasterInfo Name="MultiEventRaster">
        <FieldInfo Name="MultiEventField">
            <EventInfo EditType="Total">
                <Time>February-13-2018 08:23:11</Time>
                <Raster Name="Raster0" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-14-2018 08:23:11</Time>
                <Raster Name="Raster1" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-15-2018 08:23:11</Time>
                <Raster Name="Raster2" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-16-2018 08:23:11</Time>
                <Raster Name="Raster3" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-17-2018 08:23:11</Time>
                <Raster Name="Raster4" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-18-2018 08:23:11</Time>
                <Raster Name="Raster5" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-19-2018 08:23:11</Time>
                <Raster Name="Raster6" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-20-2018 08:23:11</Time>
                <Raster Name="Raster7" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-21-2018 08:23:11</Time>
                <Raster Name="Raster8" Field="0" Band="0"/>
            </EventInfo>
            <EventInfo EditType="Total">
                <Time>February-22-2018 08:23:11</Time>
                <Raster Name="Raster9" Field="0" Band="0"/>
            </EventInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

9. Resample.mvr

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="DTM" Filename="SmallDTM.mrr"/>
    <RasterOperationList Name="Resample">
        <Merge Enable="True">
            <Name>Merge2Resample</Name>
            <Description>Execute a Merge operation on a single file to demonstrate resampling</Description>
            <UnderviewInterpolation>Cubic</UnderviewInterpolation>
            <InterpolationNearest>False</InterpolationNearest>
            <Layer Enable="True">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Layer>
            <RasterInfo>
                <FieldInfo Name="DTM">
                    <Transform Type="Simple">
                        <Origin>664997.5,6297997.5</Origin>
                        <CellSize>20,20</CellSize>
                    </Transform>
                </FieldInfo>
            </RasterInfo>
        </Merge>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <Operation Name="Resample"/>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

This example shows how to use a Merge operation to perform a resample. The example uses a single raster source file which has a 5 metre cell size. To crystallise the resample, you need to execute a convert operation to convert the MVR to any other raster format. The convert operation will operate on the base level resolution and in the MVR this has a cell size of 20 metres.

We need to define the structure of the output raster and in this example there are three sources of this information. The first is the source raster to the merge operation. This raster provides all the default raster information. Then, we define RasterInfo in the merge operation and in here we declare the settings we wish to override. In this case, we override the Transform for the field and set a new cell size and origin. Finally, we must define RasterInfo in the VirtualRaster but in here we only include the information required to link fields or bands to raster sources – which in this case is the merge operation.

It is important to use appropriate interpolation settings – for generating underviews and for interpolating into the source buffer. Interpolation settings can be defined in three places – in the "Merge" element, in the "Operation" element, or externally (for example from the "RasterQuality" setting on the Raster ribbon in MapInfo Pro. They apply in that order, if defined. The Merge settings override the Operation and external settings and the Operation settings override external settings. The external settings are only ever used if interpolation settings are not defined in the MVR.

The raster source inputs to the merge operation must be declared in Layer elements and in this element we also define the field and band the operation will target in the source raster.

Note that this example only uses a single field and band. The merge operation has only been designed to operate on a single field and band. If you want to operate on a multi-banded field you would have to declare a merge operation for each band and you would probably have to declare the band information in detail. No example has ever been developed for this.

10. Reproject.mvr

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="DTM" Filename="SmallDTM.mrr"/>
    <RasterOperationList Name="Reproject">
        <Merge Enable="True">
            <Name>Merge2Reproject</Name>
            <Description>Execute a Merge operation on a single file to demonstrate reprojection</Description>
            <UnderviewInterpolation>Cubic</UnderviewInterpolation>
            <InterpolationNearest>False</InterpolationNearest>
            <Layer Enable="True">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Layer>
            <RasterInfo>
                <CoordinateSystem>CoordSys Earth Projection 1, 104</CoordinateSystem>
                <FieldInfo Name="DTM">
                    <Transform Type="Simple">
                        <Origin>148,-35</Origin>
                        <CellSize>Measure(1,21600),Measure(1,21600)</CellSize>
                    </Transform>
                </FieldInfo>
            </RasterInfo>
        </Merge>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <Operation Name="Reproject"/>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

This example shows how to use a Merge operation to perform a reprojection. I have added a coordinate system string to the RasterInfo element in the merge operation, which overrides the coordinate system of the source raster. In addition, I have to define a suitable Transform element with an origin and cell size. I have chosen to express the cell size as a fraction. I use 1/6 of an arc second which is about equivalent to the 5 metre original cell size.

11. Align.mvr

This example shows how to use a Merge operation to perform an alignment. To align a raster with another raster may require both a shift in the origin (and thus cell alignment) of the raster as well as a change in the cell size.

To achieve this you need to define a new Transform element in the field and declare the origin coordinate and cell size. In this example I have shifted the origin by exactly half a cell. The merge operation will use the interpolation settings defined to interpolate new values for the cell centres.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="DTM" Filename="SmallDTM.mrr"/>
    <RasterOperationList Name="Align">
        <Merge Enable="True">
            <Name>Merge2Resample</Name>
            <Description>Execute a Merge operation on a single file to demonstrate aligning</Description>
            <UnderviewInterpolation>Cubic</UnderviewInterpolation>
            <InterpolationNearest>False</InterpolationNearest>
            <Layer Enable="True">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Layer>
            <RasterInfo>
                <FieldInfo Name="DTM">
                    <Transform Type="Simple">
                        <Origin>664995,6297995</Origin>
                        <CellSize>5,5</CellSize>
                    </Transform>
                </FieldInfo>
            </RasterInfo>
        </Merge>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <Operation Name="Align"/>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

12. Clip.mvr

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="DTM" Filename="SmallDTM.mrr"/>
    <RasterOperationList Name="Clip">
        <Merge Enable="True">
            <Name>Merge2Clip</Name>
            <Description>Execute a Merge operation on a single file to demonstrate clipping</Description>
            <UnderviewInterpolation>Cubic</UnderviewInterpolation>
            <InterpolationNearest>False</InterpolationNearest>
            <TABFilename>Polygon.tab</TABFilename>
            <CellInPolygonTest>Centroid</CellInPolygonTest>
            <Layer Enable="True">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Layer>
        </Merge>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <Operation Name="Clip"/>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

This example shows how to use a Merge operation to perform a Clip to a set of polygons. To clip to all polygons in a TAB file, simply add the TABFilename element and, if desired, specify the CellInPolygonTest rule.

13. Merge.mvr

This example illustrates a merge operation that merges three rasters of varying resolution and coordinate systems in real time.

When merging rasters you generally want to specify the “bottom to top” order of the rasters so that lower resolution or lower quality rasters are overprinted by higher resolution or higher quality rasters. The “Layer” mechanism exists to allow you to achieve this. The layers are processed in the order they appear in the file, which is equivalent to “bottom to top”. Another way to achieve this is to list multiple raster files in a single raster source. In this case, the files are processed in the order in which they appear in the file.

In this example I specify a coordinate system for the MVR which is equivalent to the coordinate system of the highest resolution raster. I also choose a cell size and origin that is equal to this high resolution raster. In general, it is a good idea to make rasters align if you can because it reduces interpolation.

The data type of the source rasters differ in this example. Some are REAL4 and one in SIGNED_INT16. To make sure I get a REAL4 output, I specify the data type for the DSM band. Related to this, I make sure that UseSourceDataType is false so that all data is acquired as REAL4 and I do not see a loss of resolution.

To ensure that I get a smooth interpolation from the underviews of the low resolution rasters, I specify interpolation settings in the merge operation and use cubic interpolation in the underview generation phase and bilinear in the second phase.

If you wanted to clip the individual rasters that you were planning to merge, then you could use a merge MVR to clip each raster and then consume these MVR’s as input to the final merge operation. Another common operation is to use clipping in the merge operation to clip the rasters to a high resolution coastline or LGA polygon.

Often, the individual inputs require some data conditioning prior to merging. You can declare data conditioning objects and use them in individual layers. Another possibility is to use a Transform MVR to transform the source rasters individually (for example to apply a vertical translation and scaling to each) prior to merging.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="R1" Filename="Merge/LowRes.mrr"/>
    <Raster Name="R2" Filename="Merge/MediumRes.mrr"/>
    <Raster Name="R3" Filename="Merge/HighRes.mrr"/>
    <RasterOperationList Name="Reproject">
        <Merge Enable="True">
            <Name>Merge</Name>
            <Description>Execute a Merge operation</Description>
            <UnderviewInterpolation>Cubic</UnderviewInterpolation>
            <InterpolationNearest>False</InterpolationNearest>
            <UseSourceDataType>False</UseSourceDataType>
            <Layer Enable="True">
                <Raster Name="R1" Field="0" Band="0"/>
            </Layer>
            <Layer Enable="True">
                <Raster Name="R2" Field="0" Band="0"/>
            </Layer>
            <Layer Enable="True">
                <Raster Name="R3" Field="0" Band="0"/>
            </Layer>
            <RasterInfo>
                <CoordinateSystem>CoordSys Earth Projection 8, 79, "m", -2, 49, 0.9996012717, 400000, -100000 Bounds (0, -
100000) (2000000, 1900000)</CoordinateSystem>
                <FieldInfo Name="DSM">
                    <Transform Type="Simple">
                        <Origin>250000,150000</Origin>
                        <CellSize>2,2</CellSize>
                    </Transform>
                    <BandInfo Name = "DSM">
                        <DataType>REAL4</DataType>
                    </BandInfo>
                </FieldInfo>
            </RasterInfo>
        </Merge>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <Operation Name=" Merge"/>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

14. CalculatorXML.mvr

XML uses a number of special characters and unfortunately these are also common characters in a calculator expression. If you want to use one of these characters in a calculator expression you have to replace it with a special character sequence to ensure the XML is parsed correctly.

The special characters and the character sequences you ought to use are –

Less than	<	&lt;
Greater than	>	&gt;
Not equal	<>	<>
And	&	&amp;

Note that you can use ‘and’ as a replacement for ‘&’ in an expression. The example provided illustrates the use of these special character sequences.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="DTM" Filename="SmallDTM.mrr"/>
    <RasterOperationList Name="ClipUpper">
        <Calculator Name="ClipUpper">
            <Variable Name="DTM">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Variable>
            <Expression>cond(DTM &gt; 600,null,DTM)</Expression>
        </Calculator>
    </RasterOperationList>
    <RasterOperationList Name="ClipLower">
        <Calculator Name="ClipLower">
            <Variable Name="DTM">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Variable>
            <Expression>cond(DTM &lt; 525,null,DTM)</Expression>
        </Calculator>
    </RasterOperationList>
    <RasterOperationList Name="ClipEqual">
        <Calculator Name="ClipEqual">
            <Variable Name="DTM">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Variable>
            <Expression>cond(floor(DTM/20)*20 &lt;&gt; 600,null,DTM)</Expression>
        </Calculator>
    </RasterOperationList>
    <RasterOperationList Name="ClipAnd">
        <Calculator Name="ClipAnd">
            <Variable Name="DTM">
                <Raster Name="DTM" Field="0" Band="0"/>
            </Variable>
            <Expression>cond(DTM &lt;525 &amp; DTM &lt; 600,null,DTM)</Expression>
        </Calculator>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo Name="ClippedDTM">
            <BandInfo Name="ClipUpper">
                <DataType>REAL4</DataType>
                <Operation Name="ClipUpper"/>
            </BandInfo>
            <BandInfo Name="ClipLower">
                <DataType>REAL4</DataType>
                <Operation Name="ClipLower"/>
            </BandInfo>
            <BandInfo Name="ClipEqual">
                <DataType>REAL4</DataType>
                <Operation Name="ClipEqual"/>
            </BandInfo>
            <BandInfo Name="ClipAnd">
                <DataType>REAL4</DataType>
                <Operation Name="ClipAnd"/>
            </BandInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

15. HistTransform.mvr

This example uses the Transform operation to transform magnetic data.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="Local" Filename="Transform/Local.mrr"/>
    <Raster Name="Regional" Filename="Transform/Regional.mrr"/>
    <DataTransform Name="HistMatch">
        <Type>HistogramMatch</Type>
    </DataTransform>
    <RasterOperationList Name="HistTrans">
        <Transform>
            <DataTransformName>HistMatch</DataTransformName>
            <IndexTransform>None</IndexTransform>
            <Raster Name="Local" Field="0" Band="0"/>
            <ReferenceRaster Name="Regional" Field="0" Band="0"/>
        </Transform>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <BandInfo>
                <Operation Name="HistTrans"/>
            </BandInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

A HistogramMatch data transform is declared and in the Transform operation we consume this data transform and supply both the input raster and the reference raster. The input raster has values about a mean of 57688. The reference raster has values about a mean of 54. The output of the virtual raster is the input values, transformed via histogram matching, into the reference raster range.

To see the two rasters in the same context, you can use an MVR to merge the regional raster with the transformed virtual raster as shown in the file HistTransformMerge.mvr. Note that this merge MVR is quite simple and illustrates the minimum amount of information you need to supply to execute a merge.

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="R1" Filename="Transform/Regional.mrr"/>
    <Raster Name="R2" Filename="HistTransform.mvr"/>
    <RasterOperationList Name="Reproject">
        <Merge Enable="True">
            <Name>Merge</Name>
            <Description>Execute a Merge operation</Description>
            <UnderviewInterpolation>Cubic</UnderviewInterpolation>
            <InterpolationNearest>False</InterpolationNearest>
            <UseSourceDataType>False</UseSourceDataType>
            <Layer Enable="True">
                <Raster Name="R1" Field="0" Band="0"/>
            </Layer>
            <Layer Enable="True">
                <Raster Name="R2" Field="0" Band="0"/>
            </Layer>
        </Merge>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <Operation Name=" Merge"/>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

16. Classify.mvr

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="DTM" Filename="SmallDTM.mrr"/>
    <RasterOperationList Name="Classify">
        <Classify Name="Classify">
            <Raster Name="DTM" Field="0" Band="0"/>
            <ClassifyFailure>InvalidCell</ClassifyFailure>
            <RasterInfo>
                <FieldInfo Type="Classified">
                    <BandInfo Name="Index">
                        <DataType>UNSIGNED_INT8</DataType>
                        <NullValueType>Mask</NullValueType>
                        <BandValueType>Unique</BandValueType>
                    </BandInfo>
                    <TableInfo Name="Classes">
                        <BandCount>3</BandCount>
                        <DataCount>5</DataCount>
                        <BandInfo Name="Value">
                            <Type>Value</Type>
                            <DataType>REAL4</DataType>
                        </BandInfo>
                        <BandInfo Name="Label">
                            <Type>Label</Type>
                            <DataType>STRING</DataType>
                        </BandInfo>
                        <BandInfo Name="Color">
                            <Type>Color</Type>
                            <DataType>RGB</DataType>
                        </BandInfo>
                    </TableInfo>
                    <TableData>
                        <Data>450,"400 - 500","RGB(0,0,255)"</Data>
                        <Data>512.5,"500 - 525","RGB(0,255,255)"</Data>
                        <Data>537.5,"525 - 550","RGB(255,255,0)"</Data>
                        <Data>587.5,"575 - 600","RGB(255,0,255)"</Data>
                        <Data>625,"600 - 650","RGB(255,0,0)"</Data>
                    </TableData>
                </FieldInfo>
            </RasterInfo>
            <ClassificationMap>
                <Type>ValueRange</Type>
                <Range>400,500,0</Range>
                <Range>500,525,1</Range>
                <Range>525,550,2</Range>
                <Range>575,600,3</Range>
                <Range>600,650,4</Range>
            </ClassificationMap>
        </Classify>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <BandInfo>
                <Operation Name="Classify">
                    <UnderviewInterpolation>Nearest</UnderviewInterpolation>
                    <InterpolationNearest>True</InterpolationNearest>
                </Operation>
            </BandInfo>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

This example illustrates how to execute a classification operation in an MVR. A Classify operation is defined and consumed in the raster band structure. For illustration, I set the interpolation parameters to nearest neighbour.

The Classify operation requires a source raster which is supplied in the usual way. In addition, it is necessary to fully define the raster info because you need to have a classified field and a classification table. This information cannot be acquired from the source raster so you have to manually define it.

Firstly, you will require a single band to store the class index. This band will be an unsigned integer type (8, 16 or 32 bit depending on how many classes you require). This is the only band that a classified field can contain. All other bands are acquired from the classification table.

Now define the classification table. Each column (or band) must be defined and they can be assigned special types that are used to identify standard kinds of data in a class table. Then define the table data itself – one data item for each band in the table and one row for each row in the table.

Finally, you must declare the classification map. This is a data transform that maps from data to class index. In this example I use a simple ValueRange map where data ranges map to indices. Note that these ranges do not cover all the source data so some cells will fail to be classified and in this case I will assign them InvalidCell.

For another example, see "LC08_L1TP_089083_20191106_20191115_01_T1_QAClass.mvr" which uses an external LEG file to declare both the map and the table, considerably simplifying the MVR XML.

17. Warp.mvr

```
<?xml version="1.0" encoding="UTF-8"?>
<VirtualRaster>
    <Raster Name="BARRINGTON TOPS" Filename="Warp/BARRINGTON TOPS.mrr"/>
    <RasterOperationList Name = "WarpTopo">
        <Warp Enable = "True" Description = "Warp a raster">
            <Raster Name="BARRINGTON TOPS" FIeld="0"/>
            <RasterInfo>
                <Name>Barrington Tops 1:25000 Topographic Map Sheet</Name>
                <CoordinateSystem>CoordSys Earth Projection 1, 104</CoordinateSystem>
                <BaseTileSize>1024,1024</BaseTileSize>
                <OverviewTileSize>1024,1024</OverviewTileSize>
                <FieldInfo>
                    <Name>Topography</Name>
                    <Type>Image</Type>
                    <ValidFlagPerBand>False</ValidFlagPerBand>
                    <Transform>
                        <Type>Simple</Type>
                        <Origin>0,0</Origin>
                        <CellSize>Measure(1,30000),Measure(1,30000)</CellSize>
                    </Transform>
                    <BandInfo>
                        <Name>Color</Name>
                        <DataType>RGBA</DataType>
                        <StoreDataType>RGBA</StoreDataType>
                        <RestrictDecimals>False</RestrictDecimals>
                        <Clip>False</Clip>
                        <Transform>False</Transform>
                        <PredEncode>False</PredEncode>
                        <PredEncodeType>None</PredEncodeType>
                        <NullValueType>Mask</NullValueType>
                        <BandValueType>Standard</BandValueType>
                    </BandInfo>
                </FieldInfo>
            </RasterInfo>
            <WarpTransformType>Projective</WarpTransformType>
            <GCPList>
                <GCP>350,157,151.25,-32.125</GCP>
                <GCP>7778,157,151.5,-32.125</GCP>
                <GCP>7783,4521,151.5,-32</GCP>
                <GCP>345,4522,151.25,-32</GCP>
            </GCPList>
            <ClipRect>151.25,-32.125,151.5,-32</ClipRect>
        </Warp>
    </RasterOperationList>
    <RasterInfo>
        <FieldInfo>
            <Name>NSW Topographic 1:25000</Name>
            <Operation>
                <Name>WarpTopo</Name>
                <UnderviewInterpolation>Cubic</UnderviewInterpolation>
                <InterpolationNearest>False</InterpolationNearest>
            </Operation>
        </FieldInfo>
    </RasterInfo>
</VirtualRaster>
```

This example illustrates how to use a warp operation in a virtual raster. The source image is a topographic map that was downloaded as a PDF, converted to a PNG and then converted to an MRR. The warp operation clips out the map from the surrounding borders and locates it in space using a projective transform. For this transform, you only need the world coordinates and pixel coordinates of the four corners of the map. These coordinates are supplied in a GCP list and the clipping rectangle is specified in world coordinates. Note that in an MRR the cell coordinates increase to the north whereas in an image like a PNG they increase to the south.

18. Transparency04.mrd

```
<?xml version="1.0" encoding="UTF-8"?>
<RenderAlgorithm>
    <Raster Name="Bottom" Filename="Merge\LowRes.mrr"/>
    <Raster Name="Middle" Filename="Merge\MediumRes.mrr"/>
    <Raster Name="Top" Filename="Merge\HighRes.mrr"/>
    <ColorTable Name = "Greyscale" Interpolate = "RGB">
        <Color Red = "32" Green = "32" Blue = "32" Alpha = "255" Index = "0"/>
        <Color Red = "224" Green = "224" Blue = "224" Alpha = "255" Index = "192"/>
    </ColorTable>
    <ColorTable Name = "PseudocolorEnhanced" Interpolate = "HSL">
        <Color Red = "128" Green = "196" Blue = "255" Alpha = "255" Index = "0"/>
        <Color Red = "0" Green = "0" Blue = "255" Alpha = "255" Index = "256"/>
        <Color Red = "255" Green = "0" Blue = "0" Alpha = "255" Index = "768"/>
        <Color Red = "255" Green = "128" Blue = "128" Alpha = "255" Index = "1023"/>
    </ColorTable>
    <ColorTable Name = "Earth" Interpolate = "HSL">
        <Color Red = "128" Green = "64" Blue = "0" Alpha = "255" Index = "0"/>
        <Color Red = "0" Green = "128" Blue = "128" Alpha = "255" Index = "127"/>
        <Color Red = "0" Green = "255" Blue = "0" Alpha = "255" Index = "255"/>
        <Color Red = "128" Green = "128" Blue = "255" Alpha = "255" Index = "767"/>
        <Color Red = "192" Green = "192" Blue = "192" Alpha = "255" Index = "1023"/>
    </ColorTable>
    <DataTransform Name = "EqualAreaNL">
        <Type>EqualAreaNonLinear</Type>
    </DataTransform>
    <DataTransform Name = "Linear">
        <Type>Linear</Type>
        <RangeType>Percentile</RangeType>
        <RangeMin>0.5</RangeMin>
        <RangeMax>1</RangeMax>
    </DataTransform>
    <Shadow>
        <Enable>True</Enable>
        <Azimuth>45</Azimuth>
        <Altitude>60</Altitude>
        <Scale>50</Scale>
    </Shadow>
    <Highlight>
        <Enable>True</Enable>
        <Azimuth>315</Azimuth>
        <Altitude>60</Altitude>
        <Scale>50</Scale>
    </Highlight>
    <LayerBlendingRule>Blended</LayerBlendingRule>
    <BlendWithBuffer>True</BlendWithBuffer>
    <BufferBackgroundColor>RGBA(192,192,192,255)</BufferBackgroundColor>
    <Layer Type = "LUTColor">
        <ColorIntensity>75</ColorIntensity>
        <Component Type = "Color" Enable = "True">
            <Raster Name = "Bottom" Field = "0" Band = "0"/>
            <ColorTableName>Greyscale</ColorTableName>
            <DataTransformName>EqualAreaNL</DataTransformName>
        </Component>
        <Component Type = "Intensity" Enable = "True">
            <Raster Name = "Bottom" Field = "0" Band = "0"/>
        </Component>
        <Component Type = "Opacity" Enable = "True">
            <Raster Name = "Bottom" Field = "0" Band = "0"/>
            <DataTransformName>Linear</DataTransformName>
        </Component>
    </Layer>
    <Layer Type = "LUTColor">
        <ColorIntensity>75</ColorIntensity>
        <Component Type = "Color" Enable = "True">
            <Raster Name = "Middle" Field = "0" Band = "0"/>
            <ColorTableName>Earth</ColorTableName>
            <DataTransformName>EqualAreaNL</DataTransformName>
        </Component>
        <Component Type = "Intensity" Enable = "True">
            <Raster Name = "Middle" Field = "0" Band = "0"/>
        </Component>
        <Component Type = "Opacity" Enable = "True">
            <Raster Name = "Middle" Field = "0" Band = "0"/>
            <DataTransformName>Linear</DataTransformName>
        </Component>
    </Layer>
    <Layer Type = "LUTColor">
        <ColorIntensity>75</ColorIntensity>
        <Component Type = "Color" Enable = "True">
            <Raster Name = "Top" Field = "0" Band = "0"/>
            <ColorTableName>PseudocolorEnhanced</ColorTableName>
            <DataTransformName>EqualAreaNL</DataTransformName>
        </Component>
        <Component Type = "Intensity" Enable = "True">
            <Raster Name = "Top" Field = "0" Band = "0"/>
        </Component>
        <Component Type = "Opacity" Enable = "True">
            <Raster Name = "Top" Field = "0" Band = "0"/>
            <DataTransformName>Linear</DataTransformName>
        </Component>
    </Layer>
</RenderAlgorithm>
```

Four rendering algorithm examples are provided to illustrate how to use opacity in a rendering algorithm.

The first example, “Transparency01.mrd”, contains three LUTColor layers for a set of overlapping DEM rasters. The layer blending rule is set to “Override” which turns blending off. In this example, the layers that are on top overwrite the layers underneath.

“Transparency02.mrd” illustrates the use of the “LightTable” blending rule. To use this rule we have to set a light table blend factor which establishes the opacity for all layers in the stack. Where layers overlay each other they are blended and the colors are faded. Where layers are not overlain by other layers the color remains rich. This is the advantage of the light table rule.

“Transparency03.mrd” enables the blending rule and then sets the opacity of each layer independently. To display these algorithms in TestWindow you need to use a buffer background color of light grey and to make sure that the layers fade to this color, we turn on blending with the buffer. Note how all layers are faded regardless of whether they overlap with other layers.

“Transparency04.mrd” enables blending and then uses opacity modulation in each layer to set the opacity per-pixel rather than for the entire layer. To do this you need to enable the Opacity component, connect it to a raster data source and establish a data transform. In this example I make the areas with low terrain values transparent and the areas with high terrain opaque.

An additional rendering algorithm example is provided “LC08_L1TP_089083_20191106_20191115_01_T1_QARender.mrd”. This example renders Landsat 8 QA data using a legend to establish both the color table and the data transform.

Reference Guide

XML element names are listed, followed by the expected data type and format of the element data and then a description of the element. The following codes are used to describe data types –

UINT8, INT8	Unsigned or signed 8 bit integer
UINT16, INT16	Unsigned or signed 16 bit integer
UINT32, INT32	Unsigned or signed 32 bit integer
UINT64, INT64	Unsigned or signed 64 bit integer
REAL4, REAL8	Decimal number, 32 bit (single precision) or 64 bit (double precision)
Bool	Boolean flag written as a text string – True or False.
UTF8, UTF16, UTF32	Unicode strings in single byte, double byte and quad byte format.
String	ASCII string
Enumeration	A set of allowed values represented by unique text strings
Measure	A fractional value written as Measure(N,D) where N (numerator) and D (denominator) are UINT64

(1.1) Root element

<?xml version="1.0" encoding="UTF-8"?>	Every MVR file should contain this standard header of commands, see notes 1.1(a).
VirtualRaster	Every MVR file needs to contain a single root element declaration.

(1.2) VirtualRaster element

Enable	Bool	Enable or disable this object. By default all objects are enabled. To express the Bool variable, see notes 1.2(a).
Name	UTF8	The name of the object, see notes 1.2(b).
Description	UTF8	An optional description of this object.
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).
Version	String	Version string in “Major.Minor” format. Current version is 1.0.
InputPath	UTF8	An optional file path for input files, see notes 1.2(c).
OutputPath	UTF8	An optional file path for output files, see notes 1.2(c).
Cache2File	Bool	Cache raster data to temporary file, see notes 1.2(d).
Raster		Element containing a named raster object, see 1.3(a).
DataConditioning		Element containing a named data conditioning object.
DataTransform		Element containing a named data transform object.
RasterInfo		Element defining the structure of the virtual raster.
RasterOperationList		Element containing a named list of processing operations
RenderAlgorithm		Element containing a named rendering algorithm.

(1.3) [VirtualRaster]Raster element

Name	UTF8	The name of the object, unique and required. See notes 1.3(a).
Filename	UTF8	The filename of the raster, see notes 1.3(a).
OpenOnDemand	Bool	Open each raster when needed and close immediately, see notes 1.3(b).
OpenBaseSupport	Bool	Open each raster for Base support only, see notes 1.3(c).
DriverPreferences		Element containing driver preferences, see notes 1.3(d)

(1.4) [VirtualRaster][Raster]DriverPreferences element

GlobalDriverWarpAction	Enumeration NoAction FaillfWarped FaillfNotWarped	Deprecated
MountAsRasterFieldType	Enumeration Default Classified Image ImagePalette Continuous	Force the driver to use a specific field type. See notes 1.4(a).
UseExtCoordSys	Enumeration NoneByNone	Override the internal coordinate system with an externally supplied coordinate system, see notes 1.4(b). Do not override.

	InvalidByValid AnyByValid AnyByAny AnyBySupplied	Override invalid coordinate system by a valid external. Override by a valid external. Override by external. Override by supplied coordinate system.
UseExtRegCoords	Enumeration NoneByNone InvalidByValid AnyByValid AnyByAny	Override the internal registration coordinates with externally supplied registration coordinates, see notes 1.4(c). Do not override. Override invalid coordinate by valid external coordinates. Override by valid external coordinates. Override by external coordinates.
GlobalCompressQualitySpace	Bool	Choose compression codec that delivers higher quality (imagery) or smaller space (lossless) for overview cache files, see notes 1.4(d).
GlobalLoadTable	Bool	Force classification tables to load immediately, see notes 1.4(e).
ValidateOrUpdateStatistics	Bool	Ensure raster statistics are computed for the base level, see notes 1.4(f).
GlobalBandValueType	Enumeration Standard Discrete Unique	Specify BandValueType for each band, comma separated, see notes 1.4(g). Cell values are a point sample and can be averaged. Cell values are a mean and can be averaged. Cell values cannot be averaged.
MountViaDriver	String	Open a raster using the specified driver (supplying the driver Unique ID), see notes 1.4(f).
VehiclePictureAllowCache	Bool	Allow or prevent overview cache storage, see notes 1.4(h).
VehiclePictureSmallCellCount	UINT64	Pixel count threshold, see notes 1.4(h).
VehiclePictureIPasImage	Bool	Load ImagePalette rasters as Image type, see notes 1.4(h).
VehiclePicturePixelClip	UINT32 MinX, MinY, MaxX, MaxY	Pixel clipping rectangle, see notes 1.4(h).
AlphaIsNull	Enumeration Never PureAlpha AverageAlpha AnyAlpha PureTrans AverageTrans AnyTrans	Use the alpha band to set invalid cells, see notes 1.4(i). Do not use the alpha band to set validity. Set invalid when alpha equals zero. Set invalid when alpha <= 127. Set invalid when alpha < 255. Set invalid when alpha equals 255. Set invalid when alpha >=128. Set invalid when alpha > 0.
VehicleTIFFAllowGDALRATClassified	Bool	Allow or prevent classified field extension in TIFF driver.
VehicleTIFFNullValue	REAL8	Specify the null value for the TIFF driver.

(1.5) [VirtualRaster]DataConditioning element

Name	UTF8	The name of the object, unique and required.
Background	REAL8	Invalid values are converted to this value, see notes 1.5(a).
CapMin	REAL8	Convert all values less than this value to this value, see notes 1.5(a).
CapMax	REAL8	Convert all values greater than this value to this value, see notes 1.5(a).
Value	REAL8	Make all values equal to this value invalid, see notes 1.5(a).
Range		Element that establishes a validity range, see notes 1.5(a).
ColorRange		Element that establishes a color validity range, see notes 1.5(a).

(1.6) [VirtualRaster][DataConditioning]Range element

From	REAL8	The minimum value of the range, see notes 1.5(a).
To	REAL8	The maximum value of the range, see notes 1.5(a).
Inclusive	Bool	Invalid within or outside of the numeric range, see notes 1.5(a).

(1.7) [VirtualRaster][DataConditioning]ColorRange element

Color	UINT32	The color, see notes 1.5(a).
Tolerance	UINT32	The color tolerance, see notes 1.5(a).
Components	UINT32	The number of color components (1/2/3/4) , see notes 1.5(a).
Inclusive	Bool	Invalid within or outside of the color range, see notes 1.5(a).

(1.8) [VirtualRaster]DataTransform element

Name	UTF8	The name of the object, unique and required.
Filename	UTF8	Filename of transform, see notes 1.7(a)
Type	Enumeration	The data transform type, see notes 1.7(b).
	Pass	The data value is passed unchanged.
	PassIndex	The data index value is passed unchanged.
	PassValue	The data color value is passed unchanged.
	NBitColor	A linear transform for N bits of resolution.
	Linear	A linear transform.
	Log	A logarithmic transform.
	Sigmoid	A tuneable sigmoid transform.
	EqualArea	An equal area transform.
	EqualAreaNonLinear	A non-linear equal area transform.
	UserLinearTable	A user supplied linear transform.
	UserNonLinearTable	A user supplied non-linear transform.
	UserNonLinearPTGTable	A user supplied non-linear percentage transform.
	UserNonLinearPTLTable	A user supplied non-linear percentile transform.
	UserDiscreteValue	A user supplied discrete value transform.
	UserDiscreteRange	A user supplied discrete range transform.
	UserDiscreteString	A user supplied discrete string transform.
	BreaksAboutMean	N breaks about the mean.
	BreaksAboutMedian	N breaks about the median.
	BreaksAboutMode	N breaks about the mode.
	BreaksAboutMeanByStdDev	N standard deviation breaks about the mean.
	BreaksNatural	Jenks natural breaks.
	HistogramMatch	Histogram matching.
Reverse	Bool	Reverse the data transform (limited implementation).
Bits	UINT32	Number of bits of resolution, use with “NBitColor”.
Breaks	UINT32	Number of breaks, use with all “Breaks…” types.
ClipToNthFilledBin	UINT32	Clip the histogram at each end, see notes 1.7(c).
	Bottom, Top	Specify the number of bins to clip at the bottom at top.
RangeType	Enumeration	Bandpass clipping of the statistical range, see notes 1.7(d).
	Value	Range is specified in data values.
	Percentage	Range is specified in percentage of the statistical range.
	Percentile	Range is specified in percentile of the statistical range.
	NthFilledBin	Deprecated.
RangeMin	REAL8	Minimum range value, see notes 1.7(d).
RangeMax	REAL8	Maximum range value, see notes 1.7(d).
LimitTransformToRange	Bool	Limit the transform to the band pass range, see notes 1.7(d).
LogBase	REAL8	Log transform, logarithm base, see notes 1.7(b).
LogLimit	REAL8	Log transform, logarithm limit, see notes 1.7(b).
SigmoidK	REAL8	Sigmoid transform K value, see notes 1.7(b).
EqualAreaPercent	REAL8	Transition between EqualArea and Linear, see notes 1.7(e).
Continuous	Bool	Output continuously interpolated index values, see notes 1.7(f).
RangeDiscrete	Bool	UserDiscreteRange modifier, see notes 1.7(g).
DataIndexPosition	Enumeration	Position of the index value within a range, see notes 1.7(h).
	Bottom	Index lies at the minimum value.
	MidPoint	Index lies at the centre.
	Top	Index lies at the maximum value.
Data	Variable Inputs	Range – Index data table, see notes 1.7(j).
	Index	REAL8
	Value, Index	REAL8, REAL8
	Value1, Value2, Index	REAL8, REAL8, REAL8
	Label, Index	UTF8, REAL8

(2.1) RasterInfo element

For general comments, see notes 2.1(a).

Name	UTF8	Optional name.
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).
BaseMapSize	UINT32 X, Y	Size of the 2D base map array, see notes 2.1(c). Width and Length.
UnderviewMapSize	UINT32 X, Y	Size of the 2D underview map array, see notes 2.1(c). Width and Length.
OverviewMapSize	UINT32 X, Y	Size of the 2D overview map array, see notes 2.1(c). Width and Length.
BaseTileSize	UINT32 X, Y	Size of the base tile, see notes 2.1(c). Width and Length.
UnderviewTileSize	UINT32 X, Y	Size of the underview tile, see notes 2.1(c). Width and Length.
OverviewTileSize	UINT32 X, Y	Size of the overview tile, see notes 2.1(c). Width and Length.
GridSize	UINT64 X, Y	Number of rows and columns in the raster, see notes 2.1(d). Columns and Rows.
XMLMetaData	UTF8	An XML or text string, see notes 2.1(e).
FieldInfo		Element that defines all information in a field.

(2.2) [RasterInfo]FieldInfo element

Name	UTF8	The name of the field, optional but recommended.
Type	Enumeration Classified Image ImagePalette Continuous	The field type, see notes 2.2(a). A classified field containing an index band and a class table. An image field containing a color data band. An image field containing an index band and color palette. A field containing one or more bands of data.
Compression	Codec, Level	Compression codec and level parameter, see notes 2.2(b).
OverviewCellCoverage	Enumeration Coverage_Any Coverage_Half Coverage_Majority Coverage_Total	Rule for overview generation from underlying cells, see notes 2.2(c). If any cell is valid, create an overview cell. If at least two of four cells are valid, create an overview cell. If at least three of four cells are valid, create an overview cell. If all four cells are valid, create an overview cell.
ValidFlagPerBand	Bool	Support validity for each band or for all bands, see notes 2.2(d).
XMLMetaData	UTF8	An XML or text string, see notes 2.1(e).
Transform		Element that defines the raster transform (location information).
BandInfo		Element that defines all information in a band.
TableInfo		Element that defines all information in a classification table.
TableData		Element that defines data in a classification table.
EventInfo		Element that defines all information in an event.

(2.3) [RasterInfo][FieldInfo]Transform element

Type	Enumeration Simple Affine	The type of transform that locates the raster field, see notes 2.3(a). Located using an origin and cell size. Located using an affine transform matrix.
Origin	REAL8 X, Y	X and Y coordinates of the tile origin, see notes 2.3(a).
CellSize	REAL8 X, Y	Width and Height of the cell size, see notes 2.3(a).

(2.4) [RasterInfo][FieldInfo]BandInfo element

Name	UTF8	The name of the band, optional but recommended.
DataType	Enumeration	The data type for the band data, see notes 2.4(a).
StoreDataType	Enumeration	The data type for the band data when stored, see notes 2.4(a).
RestrictDecimals	Bool	Store decimal numbers with reduced decimal precision.
MaxDecimals	INT32	Number of decimals stored, see notes 2.4(b).
Clip	Bool	Clip data to limits when storing, see notes 2.4(c).
ClipMin	REAL8	Minimum clip limit.

ClipMax	REAL8	Maximum clip limit.
Transform	Bool	Transform data prior to storing, see notes 2.4(d).
Scale	REAL8	Transform scaling factor.
Offset	REAL8	Transform translation factor.
PredEncode	Bool	Encode data prior to storing, see notes 2.4(e)
PredEncodeType	Enumeration None PreviousColumnValue PreviousColumnLinear RunLength	Encode the data to improve compression. Do not encode. Estimate from previous value. Estimate from previous two values. Run-length encoding.
NullValueType	Enumeration None Numeric String Mask	Declares how cell validity is recorded in a raster, see notes 2.4(f). The raster has no validity information. Numeric cell value comparison. String cell value comparison. The raster stores a validity mask.
NullValue	REAL8 or String	Null value used for numeric comparison.
BandValueType	Enumeration Standard Discrete Unique	Modifier for interpolation and combination, see notes 2.4(g). Cell values are combinable and interpolated. Cell values are combinable but not interpolated. Cell values are not combinable and not interpolated.
Unit	Enumeration	Unit name or code for the band data, see notes 2.4(h).
XMLMetaData	UTF8	An XML or text string, see notes 2.1(e).

(2.5) [RasterInfo][FieldInfo]TableInfo element

Name	UTF8	The name of the table, optional but recommended.
BandCount	UINT32	The number of bands (or columns) in the table.
DataCount	UINT32	The number of rows in the table, options, see notes 2.5(a).
BandInfo		Element that defines all information about a table band.

(2.6) [RasterInfo][FieldInfo][TableInfo]BandInfo element

Name	UTF8	The name of the table band.
Type	Enumeration None Class Value Color Label Data Red Green Blue	Classification code for the table band, see notes 2.6(a). No code. Class index Primary class value Multicomponent color Class label A secondary data band Red 8 bit color component Green 8 bit color component Blue 8 bitcolor component
DataType	Enumeration	The data type of the band, see notes 2.4(a)
DataSize	UINT32	The size of the data type, see notes 2.6(b).

(2.7) [RasterInfo][FieldInfo]TableData element

Data	Variable items	A row of table data, comma delimited, see notes 2.7(a).

(2.8) [RasterInfo][FieldInfo]EventInfo element

EditType	Enumeration Partial Total	Event type classification, see notes 2.8(a). A cumulative event that modifies some data. A replacement event that modifies all data.
Time	String	Month-Day-Year HH:MM:SS e.g. February-13-2014 14:34:23

(3.0) [VirtualRaster]RasterInfo element (additional to standard elements, see notes 3.0(a))

(3.1) [VirtualRaster][RasterInfo]FieldInfo element

Operation		Element that declares the raster operation source.
RenderAlg		Element that declares the rendering algorithm source.

(3.2) [VirtualRaster][RasterInfo][FieldInfo]BandInfo element

Raster		Element that declares the raster source.
Operation		Element that declares the raster operation source.
RenderAlg		Element that declares the rendering algorithm source.

(3.3) [VirtualRaster][RasterInfo][FieldInfo]EventInfo element

Raster		Element that declares the raster source.

(3.4) [VirtualRaster][RasterInfo][FieldInfo & EventInfo]Raster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.
EventRange	UINT32 First, Last	Zero based event index range to be acquired from the source raster. Optional. First event index, last event index.
TimeRange	INT64 First, Last	Time range to be acquired from the source raster. Optional. First time, last time, see notes 3.4(a).
PrimaryRaster	Bool	Indicates the RasterInfo ought to be populated from this source.
PrimaryField	Bool	Indicates the FieldInfo ought to be populated from this source.
UnderviewInterpolation	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearest	Bool	Phase 2 interpolation setting, see notes 3.4(b).
DataConditioningName	UTF8	Name of a data conditioning object.

(3.5) [VirtualRaster][RasterInfo][FieldInfo & BandInfo]Operation element

Name	UTF8	The name of the operation or operation list must be specified.
PrimaryRaster	Bool	Indicates the RasterInfo ought to be populated from this source.
PrimaryField	Bool	Indicates the FieldInfo ought to be populated from this source.
UnderviewInterpolation	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearest	Bool	Phase 2 interpolation setting, see notes 3.4(b).

(3.6) [VirtualRaster][RasterInfo][FieldInfo & BandInfo]RenderAlg element

Name	UTF8	The name of the rendering algorithm must be specified.
PrimaryRaster	Bool	Indicates the RasterInfo ought to be populated from this source.
PrimaryField	Bool	Indicates the FieldInfo ought to be populated from this source.
UnderviewInterpolation	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearest	Bool	Phase 2 interpolation setting, see notes 3.4(b).

(4.1) [VirtualRaster]RasterOperationList element

Enable	Bool	Enable or disable this operation list.
Name	UTF8	The name of the operation list must be specified.
Description	UTF8	An optional description of the object.
Primary	Bool	Indicates this is the primary operation list in a file, see notes 4.1(a).
Calculator		Element containing a Calculator operation.
Transform		Element containing a Transform operation.
Merge		Element containing a Merge operation.
Warp		Element containing a Warp operation.
Classify		Element containing a Classify operation.

(5.1) [VirtualRaster][RasterOperationList]Calculator element

Enable	Bool	Enable or disable this operation.
Name	UTF8	The name of the operation, optional.
Description	UTF8	An optional description of the object
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).
Expression	UTF8	The calculator expression, see notes 5.1(a).
StrictNullHandling	Bool	Enable or disable string null handling in expressions, see notes 5.1(b).
Variable		Element containing a Variable for the Calculator operation.

(5.2) [VirtualRaster][RasterOperationList][Calculator]Variable element

Name	UTF8	The name of the variable, must be specified.
UnderviewInterpolation	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearest	Bool	Phase 2 interpolation setting, see notes 3.4(b).
Raster		Element that declares the raster source.

(5.3) [VirtualRaster][RasterOperationList] [Calculator][Variable]Raster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.
EventRange	UINT32 First, Last	Zero based event index range to be acquired from the source raster. Optional. First event index, last event index.
TimeRange	INT64 First, Last	Time range to be acquired from the source raster. Optional. First time, last time, see notes 3.4(a).

(6.1) [VirtualRaster][RasterOperationList]Transform element

Enable	Bool	Enable or disable this operation.
Name	UTF8	The name of the operation, optional.
Description	UTF8	An optional description of the object
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).
DataConditioningName	UTF8	Name of a data conditioning object.
DataTransformName	UTF8	Name of a data transform object.
IndexTransform	Enumeration None Scale, A, B Map, Value[]	Index transform, see notes 6.1(a). Apply no transform after the data transform. Apply a scaling from 0 – 1 to A – B. Apply a mapping from 0 – 1 to a table of values.
ClipToRange	Bool	Clip the transformed data to the index range.
ReverseRange	Bool	Reverse the index transform.
Raster		Element that declares the raster source.
TransformRaster		Element that declares the transform raster source, see notes 6.1(b).
ReferenceRaster		Element that declares the reference raster source.

(6.2) [VirtualRaster][RasterOperationList][Transform]Raster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.
EventRange	UINT32 First, Last	Zero based event index range to be acquired from the source raster. Optional. First event index, last event index.
TimeRange	INT64 First, Last	Time range to be acquired from the source raster. Optional. First time, last time, see notes 3.4(a).

(6.3) [VirtualRaster][RasterOperationList][Transform]TransformRaster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.

(6.4) [VirtualRaster][RasterOperationList][Transform]ReferenceRaster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.

(7.1) [VirtualRaster][RasterOperationList]Merge element

Enable	Bool	Enable or disable this operation, see notes 7.1(a).
Name	UTF8	The name of the operation, optional.
Description	UTF8	An optional description of the object.
UnderviewInterpolation	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearest	Bool	Phase 2 interpolation setting, see notes 3.4(b).
TABFilename	UTF8	TAB file from which polygons are acquired for clipping.
SQLColumnNameValue	UTF8 Name, Value	Query for acquiring polygon(s) from TAB file, see notes 7.1(b).
PolygonSet		Element containing a set of polygons used for clipping, see notes 7.1(g).
PolygonSetCoordSys	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b). The coordinate system of the polygons, if different to the MVR.
CellCombineRule	Enumeration First Last Minimum Maximum Merge Average	Determines how overlapping cell data are combined, see notes 7.1(c). Keep first value. Keep last value. Keep minimum value. Keep maximum value. Keep running average of values. Not implemented.
CellInPolygonTest	Enumeration Centroid Corners_Any Corners_Most Corners_All Area_Any Area_Most Area_All	Clipping rule, see notes 7.1(d). Retain if cell centroid is inside the polygon. Retain if any corner is inside the polygon. Retain if two or more corners are inside the polygon. Retain if all four corners are inside the polygon. Retain if any part of the cell is inside the polygon. Retain if most of the cell is inside the polygon. Retain if all of the cell is inside the polygon.
PIPPolygonSize	UINT32 Small, Large, Super	Not documented, see notes 7.1(e).
PIPSuperPolyArea	REAL8	Not documented, see notes 7.1(e).
PIPLeafSize	UINT32 Poly, Edge	Not documented, see notes 7.1(e).
PIPOptimisation	Enumeration None Full FewInFew FewInMany ManyInFew ManyInMany AABB SpatialPolySearch SpatialEdgeSearch MultiThread SuperPolyBlockout LookAhead	Not documented, see notes 7.1(e).
UseSourceDataType	Bool	Acquire data from source rasters in their type, see notes 7.1(f).
MergeBoundingBox	Enumeration Intersection Union RasterUnion PolygonUnion	Determines how the bounding box of the operation is computed. The default value is to use the Intersection, see Notes 7.1(h). Intersection of the bounding boxes of the raster sources and clipping polygon set. Union of the bounding boxes of the raster sources and clipping polygon set. Union of the bounding boxes of the raster sources. Bounding box of the clipping polygon set
Raster		Deprecated
DataConditioning		Deprecated
Layer		Element describing a merge layer.
RasterInfo		Element describing the structure of the output raster.

(7.2) [VirtualRaster][RasterOperationList][Merge]Layer element

Enable	Bool	Enable or disable this operation.
Name	UTF8	The name of the operation, optional.
Description	UTF8	An optional description of the object.
DataConditioningName	UTF8	Name of a data conditioning object.
DataTransformName	UTF8	Name of a data transform object – currently not supported.
Raster		Element that declares the raster source.

(7.3) [VirtualRaster][RasterOperationList][Merge][Layer]Raster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.
EventRange	UINT32 First, Last	Zero based event index range to be acquired from the source raster. Optional. First event index, last event index.
TimeRange	INT64 First, Last	Time range to be acquired from the source raster. Optional. First time, last time, see notes 3.4(a).

(8.1) [VirtualRaster][RasterOperationList]Warp element

Enable	Bool	Enable or disable this operation.
Name	UTF8	The name of the operation, optional.
Description	UTF8	An optional description of the object.
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).
AutoOrigin	Bool	The system will define an appropriate tile origin, see notes 8.1(a).
AutoCellSize	Bool	The system will define an appropriate cell size, see notes 8.1(a).
WarpTransformType	Enumeration Auto Conformal Affine Projective Polynomial_O2 Conformal_Polynomial_O2 Polynomial_O3	The style of the pixel to world transformation. Use the best appropriate. 2D polynomial 3D polynomial.
ImageYSenseDown	Bool	Pixel Y coordinates are zero at the top left, positive down.
SourceWorldCoordinates	Bool	Source coordinates are world, see notes 8.1(b).
AffineParameters	REAL8 A,B,C,D,E,F POX,POY, SHX,SHY, SX,SY, ALPHA, WOX,WOY	See notes 8.1(c). Radians counter clockwise.
ClipRect	REAL8 MinX, MinY, MaxX, MaxY	Clipping rectangle specified in world coordinates, see notes 8.1(b).
UnderviewInterpolation	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearest	Bool	Phase 2 interpolation setting, see notes 3.4(b).
GCPFilename	UTF8	Filename of an optional file containing GCP's.
GCPList		Element that contains all the GCP's.
Raster		Element that declares the raster source.
RasterInfo		Element describing the structure of the output raster.

(8.1) [VirtualRaster][RasterOperationList][Warp]GCPList element

GCP	REAL8 ImageX, ImageY, WorldX,WorldY	Ground control point definition. Source coordinates World coordinates

(8.2) [VirtualRaster][RasterOperationList][Warp]Raster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.

(9.1) [VirtualRaster][RasterOperationList]Classify element

Enable	Bool	Enable or disable this operation.
Name	UTF8	The name of the operation, optional.
Description	UTF8	An optional description of the object.
ClassifyFailure	InvalidCell Or UINT32 Value	Option to set unclassified cells invalid or to a supplied class index see notes 9.1(a).
Raster		Element that declares the raster source.
RasterInfo		Element describing the structure of the output raster.
ClassificationMap		Element describing the data to class map.

(9.2) [VirtualRaster][RasterOperationList][Classify]Raster element

Name	UTF8	The name of the raster source must be specified.	
Field	UINT32	Zero based field index to be acquired from the source raster.	
Band	UINT32	Zero based band index to be acquired from the source raster.	
EventRange	UINT32 First, Last	Zero based event index range to be acquired from the source raster. Optional. First event index, last event index.	
TimeRange	INT64 First, Last	Time range to be acquired from the source raster. Optional. First time, last time, see notes 3.4(a).	

(9.3) [VirtualRaster][RasterOperationList][Classify]ClassificationMap element

Type	Enumeration Linear Logarithmic Sigmoid LinearTable NonLinearTable DiscreteValue ValueRange InclusiveRange	Data to Class Index map type, see notes 9.3(a). Linear over a defined range. Logarithmic over a defined range. Tuneable sigmoid function over a defined range. A table spread evenly over a defined range. A table spread unevenly over a range. A table mapping from value to index. A table mapping from value range to index. A table mapping from value range to index.
LogBase	REAL8	Log transform, logarithm base, see notes 1.7(b).
LogLimit	REAL8	Log transform, logarithm limit, see notes 1.7(b).
SigmoidK	REAL8	Sigmoid transform K value, see notes 1.7(b).
Legend	UTF8	Filename of an Encom Legend file which defines the mapping.
Range		Various formats depending on transform type, see notes 9.3(a)

(10.1) [VirtualRaster]RenderAlgorithm element

Enable	Bool	Enable or disable this algorithm.
Name	UTF8	The name of the algorithm, must be declared.
Description	UTF8	An optional description of the algorithm.
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).
Version	String	Version string in “Major.Minor” format. Current version is 1.0.
RasterGHX	UTF8	Filename of a GHX file containing rendering metadata, see notes 10.1(a).
BufferBackgroundColor	Color	Background color, see 10.1(a).
RenderInvalidPixel	Bool	Render invalid pixel in a special, generally non transparent, color.
InvalidPixelColor	Color	Color for invalid pixels, if rendered, see notes 10.1(a).
ValidCellByComponentRule	Enumeration Any Primary All AllToAny PrimaryToAny	Rule that determines validity for a pixel, see notes 10.1(b). Any valid component results in a valid pixel. The primary component must be valid. All components must be valid. Two stage – “All” and if that fails, “Any”. Two stage – “Primary” and if that fails, “Any”.
LayerBlendingRule	Enumeration Override Overprint LightTable Blended	Transparency blending control, see notes10.1(c). Pixel opacity is set to 255. Pixel opacity may be acquired from the raster source. Pixel opacity is applied equally per layer. Pixel opacity may be modulated and layers will be blended.
BlendWithBuffer	Bool	Blend the pixels with the buffer background color.
LightTableBlendFactor	UINT8	Opacity, from 0 – 255, see notes 10.1(c).
ReverseRenderOrder	Bool	Reverse the rendering order of the layers.
UnderviewInterpolationColor	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b) and notes 10.1(d). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearestColor	Bool	Phase 2 interpolation setting, see notes 3.4(b).
UnderviewInterpolationIntensity	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b) and notes 10.1(d). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearestIntensity	Bool	Phase 2 interpolation setting, see notes 3.4(b).
Raster		Element containing a named raster object, see notes 1.3(a), 10.1(e).
DataConditioning		Element containing a named data conditioning object.
DataTransform		Element containing a named data transform object.
ColorTable		Element containing a named color table object.
Shadow		Element containing a named shadow object.
Highlight		Element containing a named highlight object.
Layer		Element containing a named layer object.

For information on how similar properties at Algorithm, Layer and Component scope override each other, see notes 10.1(f).

(10.2) [VirtualRaster][RenderAlgorithm]Shadow element

Enable	Bool	Enable or disable this object, see notes 10.2(a).
Azimuth	REAL4	Sun azimuth, degrees east of north.
Altitude	REAL4	Sun altitude, degrees above horizon.
Scale	REAL4	Shadow scaling 0 – 100.
CellToCellMean	REAL4	Cell statistics, use -1 to acquire automatically.

(10.3) [VirtualRaster][RenderAlgorithm]Highlight element

Enable	Bool	Enable or disable this object, see notes 10.2(a).
Azimuth	REAL4	Sun azimuth, degrees east of north.
Altitude	REAL4	Sun altitude, degrees above horizon.
Scale	REAL4	Specular scaling 0 – 100.
CellToCellMean	REAL4	Cell statistics, use -1 to acquire automatically.

(10.4) [VirtualRaster][RenderAlgorithm]ColorTable element

Name	UTF8	The name of the color table, must be supplied. See notes 10.4(a).
Interpolate	Enumeration RGB HSL	Color space in which to interpolate between colors. RGB color space. HSL color space.
Filename	UTF8	The filename of the color table, if required.
Color		Element containing a color definition.

(10.5) [VirtualRaster][RenderAlgorithm][ColorTable]Color element

Red	UINT8	Red component (0 – 255)
Green	UINT8	Green component (0 – 255)
Blue	UINT8	Blue component (0 – 255)
Alpha	UINT8	Alpha or Opacity component (0 – 255)
Red16	UINT16	Red component (0 – 65535)
Green16	UINT16	Green component (0 – 65535)
Blue16	UINT16	Blue component (0 – 65535)
Alpha16	UINT16	Alpha or Opacity component (0 – 65535)
Index	UINT32	The index of the color in the table, zero based.

(10.6) [VirtualRaster][RenderAlgorithm]Layer element

Enable	Bool	Enable or disable this layer.
Name	UTF8	The name of the layer, optional.
Description	UTF8	An optional description of the layer.
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).
Type	Enumeration LUTColor RGBColor Image Mask Contour	The layer type, see notes 10.6(a). Color + Intensity + Opacity. Red + Green + Blue + Intensity + Opacity. Image + Intensity + Opacity. Not implemented. Not implemented.
ColorIntensity	UINT8	Color – Intensity Balance, see notes 10.6(b).
ValidCellByComponentRule	Enumeration Any Primary All AllToAny PrimaryToAny	Rule that determines validity for a pixel, see notes 10.1(b). Any valid component results in a valid pixel. The primary component must be valid. All components must be valid. Two stage – “All” and if that fails, “Any”. Two stage – “Primary” and if that fails, “Any”.
UnderviewInterpolationColor	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b) and notes 10.1(d). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearestColor	Bool	Phase 2 interpolation setting, see notes 3.4(b).
UnderviewInterpolationIntensity	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b) and notes 10.1(d). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearestIntensity	Bool	Phase 2 interpolation setting, see notes 3.4(b).
Opacity	UINT8	Opacity, from 0 – 255, see notes 10.1(c).
MakeGreyscale	Bool	Convert color to greyscale.
Component		Element containing a component definition.

(10.7) [VirtualRaster][RenderAlgorithm][Layer]Component element

Enable	Bool	Enable or disable this component.
Name	UTF8	The name of the component, optional.
Description	UTF8	An optional description of the component.
CoordinateSystem	UTF8	Coordinate system defined in a MIF style string, see notes 2.1(b).

Type	Enumeration Color Opacity Intensity Red Green Blue Image Mask Contour	The component type, see notes 10.7(a). Color, used in LUTColor layers. Opacity used in all layer types. Intensity for sun shadow and highlight, for all layer types. Red, used in an RGBColor layer. Green, used in an RGBColor layer. Blue, used in an RGBColor layer. Red, used in an Image layer. Not implemented. Not implemented.
UnderviewInterpolation	Enumeration Nearest Linear CubicOperator Cubic Default	Phase 1 interpolation setting, see notes 3.4(b) and notes 10.1(d). Nearest Neighbour Bi-linear Cubic Cubic spline Use the smoothest interpolation method allowed for this data.
InterpolationNearest	Bool	Phase 2 interpolation setting, see notes 3.4(b).
ClipToRange	Bool	Handle data values outside the data transform range, see notes 10.7(b).
ReverseColor	Bool	Reverse the color table (in a LUTColor layer, Color component).
DataTransformName	UTF8	The name of a data transform object used by the component.
DataConditioningName	UTF8	The name of a data conditioning object used by the component.
ColorTableName	UTF8	The name of a color table object used by the component.
Raster		Element that declares the raster source.
TransformRaster		Element that declares the raster source for the transform.
Shadow		Element that defines sun shadow for an Intensity component.
Highlight		Element that defines specular highlight for an Intensity component.

(10.8) [VirtualRaster][RenderAlgorithm][Layer][Component]Raster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.
EventRange	UINT32 First, Last	Zero based event index range to be acquired from the source raster. Optional. First event index, last event index.
TimeRange	INT64 First, Last	Time range to be acquired from the source raster. Optional. First time, last time, see notes 3.4(a).

(10.9) [VirtualRaster][RenderAlgorithm][Layer][Component]TransformRaster element

Name	UTF8	The name of the raster source must be specified.
Field	UINT32	Zero based field index to be acquired from the source raster.
Band	UINT32	Zero based band index to be acquired from the source raster.

Reference Notes

1.1(a)

An MVR file contains XML and you can edit an MVR file in any text or XML editor. All XML files contain a standard header which tells the XML decoding library how to read the XML and a root element inside of which all XML elements are contained.

1.2(a)

A “Bool” tag indicates that a condition is true or false. You can express this in a wide variety of ways including “True/False”, “On/Off”, “Yes/No”, “Enabled/Disabled”, “Defined/Undefined” and “Valid/Invalid”.

1.2(b)

Objects may be assigned a descriptive name but this is only a requirement in certain cases. When you refer to an object from some other object then you will do so by using its’ name. For example, you will create raster source objects and then refer to these by their assigned name in other parts of the MVR. It follows that global objects ought to be assigned a unique name.

1.2(c)

In an MVR, filenames can be specified with a full path (e.g. C:\Data\Rasters\TMI.MRR”), with a relative path (e.g. “\Rasters\TMI.MRR”) or with no path (e.g. “TMI.MRR”). When the path is relative or undefined then the file will be found in a directory that is relative to the Input and Output paths, depending on whether the file is an input to some process or an output from that process. By default, the input and output paths are the same and they will be the directory in which the MVR file is located.

1.2(d)

The raster engine will request raster data from the virtual raster driver as tiles. The tile will be populated at run time, on the fly, by loading source raster data, running processing operations or executing rendering operations. Once computed the tile will be held in memory in the raster engine tile cache and subsequent requests for this tile can be quickly served without having to repopulate it. However, the tile will not be held in the tile cache indefinitely and, at some point, it will be discarded. If you turn on the “Cache2File” option then the tile data will be written into a temporary file before it is discarded. If the system requests this tile again, it can be loaded from the cache file instead of recreated. If it is expensive to create the tile data, this may result in higher performance. The temporary cache file will be deleted as soon as the MVR is closed.

1.3(a)

An MVR will usually use other rasters as source data. These rasters may be crystallised – i.e. contain data – or they may be other MVR files. All rasters that an MVR is going to consume ought to be declared in these global elements and given a unique name. Elsewhere in the MVR the raster will be referred to by this name – not by its filename.

A raster source usually refers to a single raster. In some cases it is useful to associate more than one raster file with a raster source. This can be done by including more than one “Filename” element or by using a wildcard in the filename, or both. A raster source may actually refer to hundreds of individual rasters. This is often used when employing the rendering engine. If you have many rasters of the same kind (like located images for example), then you can list them as a single raster source and use them as input to a single rendering layer definition. The rendering engine will render all the listed rasters using the layer properties.

Note that the filename can contain a full path specification, a relative path (usually subordinate to the MVR file path) or no path. See notes 1.2(c).

1.3(b)

For efficiency rasters are opened by the raster engine and held open for as long as they are needed. This incurs the cost of opening the raster once and allows raster tile data to be cached in memory. If you have a large number of rasters you can open them only when needed and then close them again immediately. Use of this option is not recommended.

1.3(c)

Rasters are opened for “Full” support by default, meaning that overview pyramids will be created and cached if they do not already exist. The MVR engine requires a populated overview pyramid to function properly. In advanced and very specific cases, you may choose to disable this for efficiency. Use of this option is not recommended.

1.3(d)

When a raster is opened a set of optional commands can be supplied that modify the behaviour of the raster driver that opens the raster. These are called Driver Preferences. It is not necessary to supply any driver preferences, but in some cases they can be very useful. Note that they are applied to all rasters in the object, so it follows that all the rasters ought to have the same structure.

1.4(a)

This applies only to the BIL and TIFF drivers and has limited functionality. If these drivers load a raster as a certain field type, you can use this option to change the field type. For example, a raster that is ImagePalette can be loaded as Image field type or you can ask for a Continuous field type.

1.4(b)

This applies to the ECW and Image drivers. The external coordinate system will be acquired from a header file, usually in TAB format. If you use the AnyBySupplied option then you should include a coordinate system in MIF format. For example AnyBySupplied,"CoordSys Earth Projection 1, 104"

1.4(c)

This applies to the ECW driver. A raster usually defines its registration coordinates internally. In some cases you can override these coordinates with ones you supply externally from a header file, usually a TAB file.

1.4(d)

The virtual raster engine requires all source rasters to have overview data pyramids and these will be generated the first time a raster is opened, if they don't already exist. This driver preference allows you to change the compression codecs used for imagery and continuous data - favouring high quality imagery and highest compression ratio for continuous data. This will result in longer processing times.

1.4(e)

This applies to the MRR driver. Large classification tables are loaded field by field, when the actual field data is requested by the raster engine. This can improve mount time for an MRR. This driver preference allows you to load all classification table data when the raster is opened.

1.4(f)

Raster statistics are required for many purposes – including rendering and some processing operations. If they have not been stored in (like an MRR) or with the raster (in a GHX file), then the raster engine will compute statistics when it needs them. However, it may not compute the base level statistics as this may take too long. It may target some lower resolution level in the overview pyramid. This driver preference makes sure that base level statistics are computed and stored for the source raster so that rendering and processing operations have access to accurate, high quality statistics.

1.4(g)

The "BandValueType" has a bearing on how the overview pyramid will be generated for a data band, and on what interpolation method is used or underview generation. Standard bands have cell values that are a point sample for the centre of the cell. We can combine cell values by averaging or other techniques and we can smoothly interpolate between cell centres. Discrete bands have cell values that are an average over the spatial extent of the cell. We can combine cell values by averaging or other techniques but use nearest neighbour interpolation to determine the value between cell centres. Unique bands have cell values that cannot be combined or averaged and can only be interpolated using nearest neighbour. A unique band behaves like a classified index band.

1.4(f)

The valid Driver Unique IDs are –

MI_ESRI_GRD	ESRI ArcGIS Grid format
MI_ESRI_ASCII_GRD	ESRI ArcGIS ASCII Grid Format
MI_ESRI_Float_GRD	ESRI ArcGIS ASCII Float Grid Format
MI_BandInterleaved_BIL	Band Interleaved (BIL) Format
MI_BandInterleaved_BIP	Band Interleaved (BIP) Format
MI_BandInterleaved_BSQ	Band Interleaved (BSQ) Format
MI_ECW_IMG	ECW and JPEG2000 Format
MI_Encom_Float_GRD	Encom Float Grid Format
MI_ERMapper_ERS_GRD	ERMapper ERS Grid Format
MI_GDAL_GENERIC	GDAL Driver Multiple Formats
MI_Geosoft_GRD	Geosoft Grid Format
MI_GeoTiff_IMG	TIFF Format, Unlocated
MI_Tiff_IMG	TIFF Format, Located
MI_Located_RawImage	JPEG and PNG Image Format, Located
MI_Unlocated_RawImage	JPEG and PNG Image Format, Unlocated
MI_MRR	MRR Format
MI_MRRPNT	MRR Point Format
MI_MRR16	MRR Version 16 Format
MI_Surfer_Binary_GRD	Surfer Float Grid Format
MI_Surfer_ASCII_GRD	Surfer ASCII Grid Format
MI_USGS_POTENT_GRD	Deprecated
MI_VerticalMapper_GRD	Vertical Mapper Grid Format

MI_VerticalMapper_GRC	Vertical Mapper Classified Format
MI_MRD	Virtual Render Driver
MI_MVR	Virtual Raster Driver

1.4(h)

This applies to the JPEG and PNG Image driver. Many images are small and users may not want to generate overview cache files for these. If the number of pixels in an image is smaller than the "SmallCellCount" threshold then no overview cache will be written, by default. The default size of this threshold is 128 MegaPixels. If, by this definition (or for other reasons), the image is "small" then it will be loaded as a single tile and no cache is created. If it is "large" then a cache file will be created. You can still prevent this if you want by disabling the "AllowCache" option.

An image that uses a color palette can be mounted as though it is an Image field rather than as an ImagePalette field. This will improve the quality of the image both in overviews and underviews because pixels in an ImagePalette field cannot be combined or interpolated.

You can elect to load a clipped rectangular window of an image by supplying the pixel coordinates of the bottom left corner and the top right corner, which can be a convenient way to clip out the border region of a map.

1.4(i)

This applies to the ECW/JPEG2000 driver and the Image (PNG/JPEG) driver. Pixels that have alpha values that meet the criteria can be set to invalid (or null). This applies when the raster contains four bands which the driver assumes are RGB and Alpha. It has to assume this as the formats do not unambiguously identify it. The settings allow alpha to contain measures of either opacity or transparency.

1.5(a)

A Data Conditioning object defines information that can be applied to real or integer data bands to "cleanse" the data. You can use it to remove or transform bad data from a raster, or to clip raster data values. The object must have a unique name which you will use to apply the object in other processing operations and rendering algorithms. The process will apply rules to determine if a value is valid or invalid and may also apply rules to convert invalid values to a valid value.

Firstly, real numbers are tested to ensure they contain a computable value (integer values are always computable). If not, they are flagged as invalid. If one or more "Value" tests are specified, then the valid input is compared to the supplied values and, if equal, is flagged invalid. If one or more "Range" tests are specified then the valid input is compared to the supplied ranges and, if within (or without, depending on the "Inclusive" flag), is flagged invalid. If a "Background" value is specified, all invalid values are now converted to this value. The minimum and maximum cap tests, if specified, are now performed on the valid value.

The "Range" element can be used to define one or more data ranges. If the input value is equal to or within that range it is flagged as invalid. If you define "Inclusive" as false, then the range is valid and anything outside of that range is invalid.

If the data conditioning object is being applied to color data values, then you can specify one or more "ColorRange" elements to invalidate a color or range of colors. This operation is performed for each color component separately, so you must specify the number of color components to be processed – either 1/2/3/4. Specify the color to be invalidated, for example in the form "RGB(255,255,255)". Then specify the tolerance, for example in the form "RGB(8,8,8)". This will invalidate all colors within a range of 8 of the key color. If "Inclusive" is false then all color values outside of this range will be invalidated.

1.7(a)

A Data Transform object is used to define a mapping from "Data" to "Index" space. In a rendering algorithm, data transforms are regularly used to convert data values to index space, so that they can then be used to acquire a color from a color look-up table. They are also used in some processing operations to transform data values. The object must have a unique name which you will use to apply the object in other processing operations and rendering algorithms.

A data transform can be defined in its entirety by an external file, in which case a "Filename" is supplied. This will generally be some variety of color look-up table, some of which can also define a data transform in addition to the color table. The full path can be supplied or a relative path and the system will also look in the MapInfo color tables directory. We support the Encom Extended LUT format, the Encom Legend format and the Vertical Mapper VCP format.

1.7(b)

This key variable defines the type of the transform. Depending on the type you choose, you may have to define other key parameters and/or the system may have to acquire key parameters from statistics for the raster that is being transformed.

"Pass" does not modify the data. In this case we assume the data is already in the range 0 – 1, which is the usual output of a transformation function. "PassIndex" is used when color mapping – in this case the data is already an integer index between 0 and 255. "PassValue" is also used when color mapping – in this case the data value is a color that requires no modification at all.

“NBitColor” defines a linear transform. The range is defined by the “Bits” parameter, which ought to be a positive integer value. It establishes a linear transform between 0 and (2^N)-1. A common application is transforming raw satellite imagery data where the data has a natural resolution of N bits.

“Linear” establishes a linear transform. By default, the transform is from the statistical minimum to the maximum of the source data but you can also set this manually.

“Log” establishes a logarithmic transform. By default, the transform is from the statistical minimum to the maximum of the source data but you can also set this manually. The transform has some special features so that it can handle values less than and equal to zero. You can set the “LogBase” parameter to set the base of the logarithm, usually 10. You can use the “LogLimit” parameter to define the point, above and below zero, at which the transform becomes linear. Values below the negative LogLimit are made positive prior to transforming.

“Sigmoid” establishes a tuneable sigmoid transform. By default, the transform is from the statistical minimum to the maximum of the source data but you can also set this manually. Define the “SigmoidK” parameter between -1 and +1 to tune the function. A value of zero results in a linear transform.

“EqualArea” defines a histogram equalised transform. The transform is constructed from histogram data acquired from statistics for the source data. You can apply bandpass limits.

“EqualAreaNonLinear” defines a histogram equalised transform. The transform is constructed from histogram data acquired from statistics for the source data. You can apply bandpass limits. This transform can be more robust that “EqualArea” where the data distribution is highly skewed or for distributions with outliers or other data anomalies.

“UserLinearTable” is a user defined transform. The user must supply a set of index values spread evenly over a defined data range. A mapping is performed from data values to data index.

“UserNonLinearTable” is a user defined transform. The user must supply a set of index values for a set of bins of variable width. A mapping is performed from data values to data index.

“UserNonLinearPTGTable” is a user defined transform where the data ranges of the bins are not specified as absolute values, but as a percentage of the statistical data range. Percentages are specified as values from 0 – 1.

“UserNonLinearPTLTable” is a user defined transform where the data ranges of the bins are not specified as absolute values, but as a percentile of the statistical data distribution. Percentiles are specified as values from 0 – 1.

“UserDiscreteValue” is a user defined transform where discrete data values (usually integer values) are mapped to data indices.

“UserDiscreteRange” is a user defined transform where discrete data ranges are mapped to data indices.

“UserDiscreteString” is a user defined transform where discrete data strings are mapped to data indices.

“BreaksAboutMean” creates N/2 bins above and below the mean, evenly spaced to the statistical limits of the data or to the range limits the user has specified. When rendering, this results in N/2 color bins above the mean and N/2 below. Similar options define breaks about the median value and the mode value. In all these cases, specify the total number of breaks using the “Breaks” item.

“BreaksAboutMeanByStdDev” creates N/2 bins above and below the mean, with bin width equal to one standard deviation of the data.

“BreaksNatural” applies the “Jenks Natural breaks” to build N bins of variable spacing.

“HistogramMatch” performs a histogram matching transform, which is different to all other transforms so far discussed. Histogram matching transforms values from one data space to another data space – it does not output index values between 0 and 1 like all the other transforms described. The input data is converted to a quantile via the input histogram, then this quantile is converted back to a data value via the reference histogram. Histogram matching can be used in the “Transform” processing operation and also in the rendering engine in which case you need to define “TransformRaster” that supplies the reference histogram.

1.7(c)

If you use statistics to build a transform, then you can elect to clip the bottom and/or top of the histogram. An example is Landsat data where the value 0 is a reserved number signalling an empty cell. Using a ClipToNthFilledBin setting of (2,1) will remove the zeros from the histogram. It will clip empty bins and filled bins until the number of filled bins encountered equals the specified value. So a clip of “2” will clip one filled bin and any empty bins up to the second filled bin. A clip value of “1” generally has no effect as the first and last bins of the histogram always contain a non-zero count of data values. This clipping occurs before any range clipping which is applied in addition to this.

1.7(d)

The Linear, Log and Sigmoid transforms require the data range to be established. This can be done by referencing the statistics or manually. To specify it manually use a RangeType of Value and set the RangeMin and RangeMax values to the data values you require. You can also set it manually by specifying a percentage of the statistical range. Percentage values range from 0 to 1. You can also set it

manually by specifying a percentile range. In this case, the data histogram is used to determine the data values associated with the percentile values. Percentage values range from 0 to 1. Note that this is applied after bin clipping (see notes 1.7(c)).

The EqualArea and EqualAreaNonLinear transforms require a histogram to build the data transform and this can also be clipped to a range using the RangeType command. Generally, the histogram will be regenerated over the clipping range prior to being used to build the transform. This can be prevented by setting the LimitTransformToRange item to false (this defaults to true).

1.7(e)

When you define an EqualArea style transform you get the best possible color distribution across the data. This often results in very high color contrasts. You can tone down this contrast by specifying a transition factor for the EqualArea transform which scales it back to a standard linear transform. A value of 1 corresponds to 100% EqualArea and a value of 0 corresponds to 0% EqualArea (linear).

1.7(f)

A data transform can be thought of as a curve defined by a series of discrete values. The Continuous item, when set to true, results in the data transform being interpolated between the discrete values. When rendering, this may result in a smooth change in colors. If you want a set of discrete values output then turn this off. By default, this option is enabled.

1.7(g)

In a UserDiscreteRange transform the user will supply a set of ranges from Minimum to Maximum. If a data value lies within this range it is mapped to the supplied index. By default, the data value must be >=Minimum and <Maximum in expectation that ranges will touch (but not overlap). By setting RangeDiscrete to true, you can change this rule to >=Minimum and <=Maximum. The default is false.

1.7(h)

All of the transforms that require you or the system to supply a table of data ranges and a data index associated with that range allow you to specify where in those data ranges the index value lies. By default it lies at the top of the range, but you can also specify that it lies at the bottom or centre of the range.

1.7(j)

This item is used when you specify a transform type of either UserLinearTable, UserNonLinearTable, UserNonLinearPTGTable, UserNonLinearPTLTable, UserDiscreteValue, UserDiscreteRange or UserDiscreteString.

For those transform types, you need to supply a table of data and what data you supply depends on the transform type.

For UserLinearTable you supply the Index for each bin. The bins are all equal width and distributed equally over the data range.

For UserNonLinearTable you supply the bottom value and Index for each bin.

For UserNonLinearPTGTable and UserNonLinearPTLTable you supply the bottom value as a percentage or percentile (0 – 1) and Index for each bin.

For UserDiscreteValue you supply the value and Index for each bin.

For UserDiscreteRange you supply the bottom and top values and Index for each bin.

For UserDiscreteString you supply the string label and the Index for each bin.

2.1(a)

The RasterInfo element defines the structure of a raster including such things as the fields and bands, the coordinate system and the cell size etc. An MVR is required to have a RasterInfo element, and in addition to the standard properties it will have some special properties – see notes 3.1 onwards. RasterInfo elements may also appear in processing operations where it is necessary for the processing operation to define the structure of the output raster.

Fortunately, it is not necessary to completely define a RasterInfo element in the vast majority of cases. What data you do not explicitly provide, the system will try to deduce from the properties of the rasters that are sources for the MVR. In practice, it is usually enough to provide the minimum amount of information - i.e. what raster source, processing operation or rendering algorithm each field or band in the MVR is attached to (and from which it acquires its data).

2.1(b)

The coordinate system ought to be defined in a MIF style string. For a definition of how these strings are constructed, refer to the MapInfo Pro documentation. Some useful coordinate system strings include –

Geodetic WGS84: CoordSys Earth Projection 1, 104
Popular Visualisation CRS: CoordSys Earth Projection 10,157,"m",0

2.1(c)

In an MVR and an MRR there are a variety of maps that store information about data tiles. The size of these maps makes no difference to the operation of the maps, but it may affect the efficiency of the map. In general, there is no good reason to define any map size information.

You can also control the size of the base, overview and underview tiles. The default tile size is 1024 x 1024 cells. You should always choose a tile size that is an integer power of 2. I do not recommend tile sizes smaller than 64x64 or larger than 2048x2048. A reason you might choose to use a tile size smaller than default is to maximise the efficiency of any multithreaded process that is consuming the MVR and is accessing MVR data by tile (for example a multithreaded rendering engine). In this scenario, choosing a smaller tile can reduce the amount of time threads spend queueing to acquire access to a tile.

2.1(d)

Neither MRR nor MVR have a fixed size. This information is mainly used for legacy rasters and grids. In some scenarios, if the expected size of the raster is known, this information can be used to optimise map and tile sizes.

2.1(e)

This element contains metadata which is expected to be in XML string format, although this is not enforced. In fact you can store any text in this element. Also, if you supply a filename (and nothing else) then the contents of this file will be loaded and stored in the metadata string. In this scenario, the system expects that the file will contain ASCII or XML data only.

2.2(a)

MVR and MRR use the field type to declare to users exactly what the field contains and how it ought to be interpreted. A Continuous field contains one or more bands of data of any type and is the most general kind of field type. An Image field contains a single color data band. An ImagePalette field is also an image, but it contains a single integer index band and a color palette which is exposed as virtual bands. A Classified field contains a single integer index band and a classification table whose columns are exposed as virtual bands.

2.2(b)

An MVR is not written to disk and so the compression settings for the field have no effect. These are used when writing formats like MRR or TIFF which are able to use a variety of different compression codecs to compress the data prior to storage on disk. There are a large number of compression codec strings. Commonly used strings are “None”, “Zip” and “LZMA” (use level 0 to 9), “PNG” and “JPEG” (use level 0 to 9).

2.2(c)

This rule does not apply to MVR. A cell in an overview level is populated by averaging the four cells that underlie it in the next level down (which may be the base level or a previous overview level). By default, if two or more of these cells are valid then a valid overview cell will be populated. One common use of this rule is to relax it so that a valid overview cell is created if one or more cells are valid. When rendering such a raster, you will always see content as you zoom out. Cells will never disappear.

2.2(d)

In a multi-banded Continuous field you can improve storage efficiency by having a single validity flag for all bands rather than for each band. For some data (like hyperspectral imagery) this may be appropriate and lead to storage efficiencies.

2.3(a)

The Transform element is used to locate a raster field. Originally, this was designed to use either a Simple transform, an Affine transform or a non-linear transform defined by ground control points. However, you should use a MVR containing a warp operation if you wish to use an Affine or GCP transform. We now only support the Simple transform, for which you need to define the tile origin and the cell size.

The tile origin coordinate is the bottom left corner of tile (0,0). Often, it will also be the bottom left corner of the bottom left cell of the raster, but there is no requirement for this. A raster is a sparse arrangement of tiles that can lie in any of the four quadrants surrounding the tile origin. The cell size is specified for the base resolution level as width and length.

You can specify these values as comma separated real values or as comma separated fractions. Express a fraction as Measure(Numerator, Denominator) where Numerator and Denominator are integer values. For rasters that have odd cell sizes, like a geodetic raster with a cell size of one arc second for example, expressing the cell size as a fraction is advised. One arc second would be expressed as Measure(1, 3600).

2.4(a)

The DataType enumeration contains many possible values, only some of which are listed below -

BIT1, BIT2, BIT4
UNSIGNED_INT8, UNSIGNED_INT16, UNSIGNED_INT32, UNSIGNED_INT64
SIGNED_INT8, SIGNED_INT16, SIGNED_INT32, SIGNED_INT64,
REAL4, REAL8
COMPLEX_REAL4, COMPLEX_REAL8

DATETIME_CPP
STRING, STRING_UTF8, STRING_UTF16, STRING_UTF32
RED, GREEN, BLUE, GREY, ALPHA
RGB, RGBA, HSI, HSIA, HSL, HSLA, HSV, HSVA
MINISBLACK1, MINISBLACK2, MINISBLACK4, MINISBLACK8
MINISWHITE1, MINISWHITE2, MINISWHITE4, MINISWHITE8

The DataType determines what data type will be used in a tile in memory. For example, a REAL4 band will contain 32 bit floating point real numbers. The StoreDataType is generally the same as the DataType. It only differs when you wish to apply a transform to the data for efficient storage. For example, you may have elevation data in REAL4 type but store it in SIGNED_INT16 and use a translation and scale to transform the data as it is stored into, and read from, the tile on disk.

2.4(b)

When storing decimal numbers to disk it can be advantageous and appropriate to reduce the number of decimal points of precision in the data. For example, if you are storing elevation you might choose to store 3 decimal points of precision, which correlates to 1mm vertical resolution. When you are using a lossless compression codec, data with reduced decimal precision compresses to a smaller binary.

2.4(c)

When storing data you can apply a clip. Any values below the clip minimum are changed to the clip minimum and any values above the clip maximum are changed to the clip maximum, and then stored. This is primarily used to support some legacy raster formats.

2.4(d)

A Transform can be applied when storing tile data to disk. On loading the tile from disk, the reverse transform will be applied. The main application for this is to transform real data to integer for more efficient storage.

When loading from disk: Data = (Stored * Scale) + Offset
When saving to disk: Stored = (Data - Offset) / Scale

2.4(e)

Data can be encoded prior to storing on disk. The only reason to do this is to achieve higher compression ratios. For integer data, predictive encoding is lossless and the data stream will be recovered without any loss of data. For decimal data predictive encoding is not lossless and it is not recommended. Run-length encoding can be efficient for rasters that have large runs of invariant data values.

2.4(f)

Rasters can have valid and invalid (or null) cells. MRR and MVR record cell validity using a mask. Legacy rasters generally use a numeric value comparison.

2.4(g)

The BandValueType is used to modify the way overview cells are generated and also to modify what interpolation methods can be used on cell data. A Standard cell value is an estimation of a continuous field at the centre of the cell. Cell values can be averaged and combined in other ways and smooth interpolation methods are permissible. A Discrete cell value is an average over the spatial extent of the cell. Cell values can be averaged and combined in other ways but smooth interpolation methods are not used. A Unique cell value is generally some kind of bit mask or other coded value. Cell values cannot be averaged or combined and interpolation methods are not used.

2.4(h)

The unit code can be defined numerically (supply a unit code number compatible with MapInfo Pro MIF style or PRJ style coordinate strings) or by name or by abbreviation. Here is a list of the common units listing the ID then the code number, name and abbreviation.

Undefined(-1)	"Undefined"	"raw"
Miles(0)	"miles"	"mi"
Kilometers(1)	"kilometers"	"km"
Inches(2)	"inches"	"in"
Feet(3)	"feet"	"ft"
Yards(4)	"yards"	"yd"
Millimeters(5)	"millimeters"	"mm"
Centimeters(6)	"centimeters"	"cm"
Meters(7)	"meters"	"m"
USSurveyFeet(8)	"US survey feet"	"survey ft"
NauticalMiles(9)	"nautical miles"	"nmi"
Links(30)	"links"	"li"
Chains(31)	"chains"	"ch"
Rods(32)	"rods"	"rd"
Degree(64)	"degree"	"deg"

ArcMinute(65)	"arcminute"	"arcmin"
ArcSecond(66)	"arcsecond"	"arcsec"
MilliArcSecond(67)	"milliarcsecond"	"mas"

There are a wide variety of other units and I have listed their names only – Microseconds, Milliseconds, Seconds, Minutes, Hours, Days, Weeks, Years, dB, dBm, dBW, dBuV/m, Radians, Percent, Degrees (dBP), Calls / hr / sq km, Msgs / hr / sq km, Erlangs / hr /sq km, Simultaneous Calls / sq km, Erlangs, Bits / Cell, KBits / sq km, MBits / sq km, Events / Sec, Kbps, Kbps / sq km / Floor, Subscribers, Subscribers / sq km, Subscribers / sq km / Floor, Erlangs / sq km, Erlangs / sq km / Floor, Mbps, Bits / Sec / Hz, Kbps / sq km, Kbps / MHz, Calls, kilometres, millimetres, centimetres, metres, degrees, Kbps/sq km/Floor, Subscribers/sq km, Subscribers/sq km/Floor, Erlang, Erlangs / hr / sq km, Erlangs/sq km, Erlangs/sq km/Floor, Bits/s/Hz, Kbps/sq km.

2.5(a)

A Classification table is a spreadsheet of data with M columns (or bands) and N rows. You do not need to specify the number of rows but if you do, and the number of rows in the table differ from the expected number, a warning will be issued.

2.6(a)

Some bands in a class table may have a special classification code which may used to identify important bands in rendering and processing operations. A zero based index will be used to link cell values to table rows. In addition to this you can record a “Class Index” in the table itself, but this is purely for decoration. Most classes will have a Label field and a Color field, or color defined by three color component fields. Some simple class tables will also have a primary data value field.

2.6(b)

For some data types it is necessary to define the length of the data object in bytes. For example, if you use a FIXED_STRING data type then you ought to declare the length of the string.

2.7(a)

The Data element ought to contain a data value for every band (column in the table) for an entire row. Add a new Data item for every row in the table.

2.8(a)

An Event occurs at a time and if a raster contains multiple events then this constitutes a time line. Each event will contain data which is stored in the raster. This data does not overwrite any existing data – it simply accumulates more data in the raster. When you request data from the raster engine you can do so for a particular time, or over a range of times. The most common request is for data from the beginning of time, up to the latest time.

When a request for data over multiple events is made, the data will be accumulated with a “painters” algorithm. This means data in later events will overprint data from earlier events. If there are a large number of events this can be inefficient. One way to improve efficiency is to distinguish between events that are cumulative (Partial) and events that represent a total replacement of the data (Total). When you request data over a time range, the system will not look back past the first Total replacement event, because data prior to that event has no relationship to data after it by definition.

3.0(a)

The RasterInfo element declared in a VirtualRaster root element must declare how each field and band of the virtual raster will acquire its raster data. This will either be another raster, a raster operation list or a rendering algorithm. You can declare this source at the FieldInfo level, in which case there is no requirement to declare the bands because they will be acquired from the field source. Alternatively, you can declare a different source for each band.

3.4(a)

Enter time in a C++ time_t format which is a 64 bit signed integer value such as might be acquired from the mktime() function. You can also use a standard time description in the format “Month-Day-Year HH:MM:SS”. An example is "February-13-2014 14:34:23, November-27-2015 08:01:44".

3.4(b)

Both the virtual raster engine and the rendering engine acquire raster data in blocks that are often tile aligned. This operation requires a raster block to be populated from a source raster where the cell size and cell alignment might differ (and even the coordinate system may differ). The system tries as hard as possible to avoid interpolating data when it populates the destination raster, but in many circumstances it is impossible to avoid. In that scenario, there are up to two interpolation phases. In the first phase, interpolation may be used to generate one or more underview tiles for a source raster. This uses an established set of methods that range from nearest neighbour (which does not do any interpolation) to cubic spline (which is very smooth). The second interpolation phase is used when the cell centres of the source raster do not align with the cell centres of the destination raster. In this phase you can use either nearest

neighbour (by setting InterpolationNearest to true) or bi-linear interpolation (the default). As a general rule, if you are using nearest neighbour to generate underviews then you ought to use nearest neighbour in phase 2. In all other cases use bi-linear in phase 2.

4.1(a)

An operation list contain one or more operations which will be executed, if enabled, in the order in which they are listed in the file. For a virtual raster, the list should only contain a single operation. If you want to execute multiple operations in a virtual raster then you can do it by chaining together virtual rasters.

The primary operation list is the first operation that will be executed when run through the RasterProcessor interface. This feature is used to nest and structure operation lists but is not used in virtual rasters.

5.1(a)

For comprehensive information about calculator expressions that are permissible, see the MapInfo Pro Advanced documentation. In an XML file it is important to note the following restrictions. XML does not allow some special characters, even inside a string literal. These include –

& - use 'and' or '&'
&& - use 'and' or '&&'
> - use '>'
< - use '<'
<> - use '< >'

The calculator expression should not make use of any band array syntax (like Grid[2]). This syntax is not supported in a calculator operation (unlike in the Calculator tool in the UI). Instead, use variables to refer to all data sources and refer to those explicitly. Also, unlike the Calculator tool, this calculator operation is not multi-banded so you have to define an expression for each band that you want processed. In this operation a "Variable" represents a raster source (using a specified field, band and time). We do not support constants as variables so these ought to be entered as numeric values in the expression.

At this time, the calculator operation does not support raster statistics functions (Rmin, Rmax etc.).

5.1(b)

When strict null handling is enabled, any simple expression that includes an input that is null will generate null as the output. If this option is disabled then some simple expressions will still return a valid value, even if one or more of the inputs to the expression is null. For example, if you add two rasters together "A + B" then you will get a valid result if either A or B is valid with strict null handling disabled. In other words the expression "A + B" it will return either the intersection of the rasters or the union, depending on this setting.

6.1(a)

The transform operation takes a raster and applies a Data transform to it and then an Index transform and emits the transformed values. This might be an end in itself, or the virtual raster may then be fed into a different virtual raster operation or rendering algorithm.

The operation firstly acquires raster source data then applies a data conditioning operation, see notes 1.5, if it has been defined (by providing the name of a globally declared data conditioning object).

The conditioned data is then transformed to an index value between 0 and 1 (but not clamped to this range) by the data transform, see notes 1.7. The user must supply the name of a globally declared DataTransform object.

The index values are then transformed by the IndexTransform. You can elect to scale the index values to a new data range or you can use a data table to transform the index values to a set of defined values. The index values, between 0 and 1, and mapped linearly to the entries in the table and no interpolation is performed.

6.1(b)

If a TransformRaster is supplied, then the statistics of this raster will be used to build the data transform instead of the statistics of the actual source raster. This only has an effect if you are using a data transform type that requires statistics to define its characteristics. This capability is often used when rendering where you have multiple small raster that you want colored the same way as a regional raster. In that scenario you supply the regional raster as the transform raster to get a common data – color mapping for all the rasters being rendered.

6.1(c)

A ReferenceRaster definition should only be supplied when using the HistogramMatching data transform type. In this scenario, we use histogram matching to transform values from the source raster into values in the range of the reference raster. The system will acquire histograms for both the source and reference rasters in order to perform the transformation. As this transformation emits values that are in the range of the reference raster, you should not specify an IndexTransform.

7.1(a)

A Merge operation can be used to perform a variety of different processing operations and is not limited to just merging. For example you can use it to perform a raster clipping operation or a raster reprojection – with or without any raster merging.

You will need to provide a RasterInfo element to define the structure of the raster that is produced by the merge. You need to provide one or more Layer elements that point to raster sources that will be merged (there is no requirement to operate on more than one raster).

The output raster may have a cell geometry that is different to any of the input rasters, so interpolation settings are important. These will be employed when acquiring source data for the merge.

7.1(b)

If a TABFilename is supplied then all of the polygons will be loaded from it and the merge operation will be clipped to this polygon set.

If, in addition, a query is supplied then you need to supply a column name and a data value. The SQL query “SELECT * FROM <Table> WHERE <Name> = <Value>”. This query allows you select a subset of the polygons in a file, where some column value matches the required value.

7.2(c)

When merging, you can expect that source cells will overlap and the destination cell will need to be populated from a stack of source cells. In this case, a rule is required to determine the output cell value. Note that the true average value of all source cells option is not implemented.

7.2(d)

The clipping rule gives you precise control over how the algorithm determines if a cell is inside a clipping polygon or not, but the choice may affect the performance of the clipping algorithm. The “Centroid” option tests whether the point at the centre of each cell is inside or on the edge of the polygon – it is the fastest option. The three “Corners…” options test the four corners of each cell and count the number of corners inside the polygon. The “Area…” options divide the cell into nine sub-cells and tests the centroids of those sub-cells. The rules are triggered if one sub-cell, 5 or more, or all sub-cells are inside the polygon.

7.1(e)

The algorithm that tests for points inside polygons has a number of advanced settings which can be controlled via these XML elements. At this time, they are undocumented.

7.1(f)

When you are merging rasters the source rasters may have different data types. For example, you may be merging a terrain raster that uses REAL4 and a regional terrain raster that uses SIGNED_INT16. If you set “UseSourceDataType” true, then data will be acquired from the source rasters in their native type. This may result in a loss of smoothness and detail, especially if underviews are employed. Otherwise, data is acquired in an appropriate floating point type and interpolation may be smoother.

7.1(g)

To clip to a polygon set you usually specify the TAB file containing the polygons. If the polygons are in a different coordinate system to the virtual raster, then they will be reprojected accordingly. Alternatively, you can define the polygon set using XML but in this case the polygons are expected to be in the same coordinate system as the virtual raster and no reprojection is performed.

The PolygonSet contains one or more polygons, each of which contain one or more holes. There is no requirement to close the polygons or holes as this is done automatically when the polygons are spatially indexed. The XML has the following structure –

```
<PolygonSet  version = "0" polygonCount = "2">
    <polygon boundaryPntCount = "4" holeCount = "1">
        <point x="0" y="0"/>
        <point x="0" y="10"/>
        <point x="10" y="10"/>
        <point x="10" y="0"/>
        <hole holePntCount="4">
            <point x="4" y="4"/>
            <point x="4" y="8"/>
            <point x="8" y="8"/>
            <point x="8" y="4"/>
        </hole>
    </polygon>
    <polygon boundaryPntCount = "4" holeCount = "0">
        <point x="20" y="0"/>
        <point x="20" y="10"/>
        <point x="30" y="10"/>
        <point x="30" y="0"/>
    </polygon>
</PolygonSet>
```

7.1(h)

The bounding box of an MVR that uses a Merge operation is usually the intersection of the union of the bounding boxes of all the raster sources and the bounding box of any clipping polygon set that is defined. If desired, you can change this to the union of these two datasets or to either of the two datasets alone.

8.1(a)

The Warp operation is designed to apply a warping transformation to an unlocated image to locate it in world coordinates. MVR gives you the option to apply this on the fly rather than create a new raster that duplicates the original image.

You need to supply a RasterInfo element to describe the structure of the output raster. However, you can use the AutoOrigin and AutoCellSize options to relieve yourself of the requirement to define the output raster tile origin and base resolution level cell size.

8.1(b)

Source coordinates can be supplied as Cell, Pixel or World coordinates. By default they are cell coordinates, but if ImageYSenseDown is enabled then they will be Pixel coordinates. They can also be supplied in world coordinates (you can warp a raster that is already located to relocate it).

You can clip the image to a rectangle, for example to remove the outer borders from a scanned map. This clipping is performed after the warp so supply the clipping rectangle coordinates in world coordinates.

8.1(c)

If you use an Affine transform you have the option to declare the affine parameters and in this case you do not have to supply GCP's. If you do not declare affine parameters then they will be determined automatically from the GCP's.

Affine transforms are defined by six parameters (A,B,C,D,E,F) and the transformation from source (pixel) to destination (world) coordinates is given by this equation -

WX = PX*A + PY*B + C
WY = PX*D + PY*E + F

These parameters are computed using the following series of transforms -

1. Translate the anchor pixel to zero

[1 0 -POX]

[0 1 -POY]

[0 0 1]

2. Scale from pixel to world (SX = SY = 1 by default)

[SX 0 0]

[0 SY 0]

[0 0 1]

3. Shear in X and Y (SHX = SHY = 0 by default)

[1 SHX 0]

[SHY 1 0]

[0 0 1]

4. Rotate (anticlockwise alpha about the anchor)

[cos -sin 0]

[sin cos 0]

[0 0 1]

5. Translate the anchor to world coordinates

[1 0 WOX]

[0 1 WOY]

[0 0 1]

You can combine all these transforms into the following formulas -

A = SX.cos - SX.SHY.sin
B = SY.SHX.cos - SY.sin
C = WOX - (POX*A+POY*B)
D = SX.sin + SX.SHY.cos
E = SY.SHX.sin + SY.cos
F = WOY - (POX*D+POY*E)

If you declare the affine parameters then you can either supply the six parameters (A,B,C,D,E,F) or the nine parameters described in the matrices listed.

9.1(a)

The Classify operation will take a numeric source raster and, via a classification map that maps source values to destination class indices, output a classified raster. You have to supply RasterInfo to declare the structure of the output raster and this ought to include a classification table with all of the class data. You also have to supply a map that will map data values to zero based class indices (which point to rows in the classification table).

If a data value cannot be successfully mapped to a class index, then you can choose whether to mark the cell as invalid or assign a special class index (which needs to be specified and needs to be part of the classification table).

9.3(a)

For a Linear and Logarithmic transform you only need to supply one Range item with the minimum data value and maximum data value of the range and the number of indices spread over that range.

For a LinearTable the first Range item ought to specify the minimum data value and maximum data value of the range. In the next Range item (and spread over as many as you want), list the index vales that each equi-spaced range maps to.

For a NonLinearTable list the minimum value of each range followed by the index – one per item. The final item should be the maximum value with a repeat of the top index.

For a DiscreteValue table just list the value followed by the index, one per item.

For a DiscreteRange and InclusiveRange table list the minimum and maximum values of the range followed by the index, one per item. An inclusive range maps values >=min and <=max to the index.

10.1(a)

A rendering algorithm invokes the rendering engine to render one or more layers, each of which can acquire data from multiple raster sources. The rendering engine renders to a bitmap buffer. When invoked from a virtual raster, the buffer is passed to the virtual raster engine as raster source data. Usually, rendering algorithms will be included in virtual rasters as image fields.

Rendering algorithms have a number of default configurations (although these are generally not applicable when used in virtual rasters). If you supply a single raster source and no layer information, a default algorithm will be generated for that raster that is most appropriate for it. If you supply a GHX filename and no layer data then an algorithm that utilises the rendering metadata in the GHX file will be generated.

All rendering occurs in a pixel buffer. This buffer is first cleared with the buffer background color – usually black or white. When you specify a color in the XML you can use a variety of formats, including:

- A text string – WHITE, LIGHTGREY, GREY, DARKGREY, BLACK, RED, MAROON, YELLOW, OLIVE, GREEN, DARKGREEN, CYAN, TEAL, BLUE, DARKBLUE, MAGENTA, PURPLE, BROWN
- A 32 bit unsigned integer representing 0xAABBGGRR
- A color component macro with either 8 color (0 – 255), 16 bit color (0 – 65535) or floating point color (0 – 1) below:
- RGB(R,G,B) 8 bit
- RGB(R,G,B) Float 0 - 1, auto-detect but use RGBF instead
- RGB(R,G,B,A) 8 bit + opacity, used in TAB files
- RGBA(R,G,B,A) 8 bit + opacity
- RGBA(R,G,B,A) Float 0 - 1 + opacity, auto-detect but use RGBF instead
- RGBF(R,G,B) Float 0 - 1
- RGBTF(R,G,B,T) Float 0 - 1 + transparency
- RGBAF(R,G,B,A) Float 0 - 1 + opacity
- RGB16(R,G,B) 16 bit
- RGBT16(R,G,B,T) 16 bit + transparency
- RGBA16(R,G,B,A) 16 bit + opacity

10.1(b)

Every layer is composed of one for more components, and each of the enabled components contributes to the coloration of each pixel. The ValidCellByComponentRule determines what components have to produce a valid result in order for the pixel to be colored. If the rule is not satisfied, then the pixel is invalid (and in most cases this results in it being transparent).

This rule generally only comes into play when the rasters being used in each component are different. For example, you may be generating color from a scanned map image and intensity from a large scale DTM raster. If you use the "All" option then only the pixels that lie in the intersection of those two rasters will be valid. If you use the "Any" option then the pixels that cover the union of those rasters will be valid.

The "Primary" option indicates that the primary component in each layer must be valid. For a LUTColor layer this is the Color component. For an Image layer it is the Image component. For an RGBColor layer it is the Red, Green and Blue components. If the primary component in a layer is disabled then the Intensity component becomes primary.

The "AllToAny" and "PrimaryToAny" are advanced options that activate a multi-stage rendering pipeline where the engine renders using the strictest rule first, through to the least strict rule. AllToAny renders using All, then Primary, then Any. PrimaryToAny renders using Primary and then Any. Using either of these rules results in lower rendering performance as layers are rendered two or three times. In the second (and third) pass, pixels are only rendered if they have not already been rendered by a previous pass. In other words, the "best quality" pixel is retained. This only works within a layer and this setting can be controlled for each layer. It is applicable if you have raster sources that contain multiple actual rasters, some of which may be overlapping.

10.1(c)

In the rendering engine we usually refer to Opacity which is the inverse of Transparency. It is usually expressed as an integer from 0 to 255 or a decimal from 0 to 1. When opacity is 255 then the pixel is opaque and when it is 0 it is transparent. The Alpha component of a color is opacity, so a value of 0 in the alpha component indicates the color is transparent and a value of 0xFF (255) indicates it is opaque.

Within a layer you can use the Opacity component to modulate the opacity of a layer. When you have multiple overlapping layers, you also have to consider how the opacity of those layers will interact. Even with a single layer, you have to consider how it will blend with the buffer background color.

To disable all blending and opacity modulation, use the Override option. When this option is used all pixels will be emitted as 100% opaque. Layers that overlap will paint over any layers lower in the stack that are underneath.

The Overprint option also disables blending and opacity modulation, but it does allow any existing alpha values to be passed through to the output. For example, if you are rendering a 32 bit image that has variable alpha values, these will be copied through to the output buffer. For LUTColor and RGBColor layers, Overprint has no effect and behaves the same as Override. It is only for Image layers that it allows existing alpha values to pass through unchanged.

LightTable is used in conjunction with the LightTableBlendFactor which establishes the opacity of all of the layers in the algorithm. The layer data is not blended with the background color so that the color intensity of the layers is enhanced. Even though layers are blended, the opacity of the output pixels will still be 100%.

To use the Opacity component in a layer and enable opacity modulation, you must set the LayerBlendingRule to Blended. In all other scenarios, the opacity component will be ignored. Now, you can set the opacity of each layer independently using the Opacity item in the Layer element. Or, you can enable opacity modulation using an Opacity component. By default, the layers will be blended with the buffer background color, but you can choose to disable this. In this mode, the opacity of the output pixels will be a product of the blended layers.

10.1(d)

At the algorithm level you can control interpolation settings (see notes 3.4(b)) for color components and the intensity component separately. In general, you will want to use smooth interpolation for the intensity component regardless of how you are interpolating color. Note that these settings are also available at layer scope and component scope. For interpolation settings, algorithm settings override layer settings and layer settings override component settings.

10.1(e)

Although Raster sources can be defined at algorithm scope, in a virtual raster these will generally be defined at the VirtualRaster scope. On the other hand, DataConditioning objects and DataTransform objects are often defined locally as they are specific to the algorithm.

10.1(f)

In a rendering algorithm there are a variety of properties that can be set at multiple levels of scope – including in the Algorithm (the top level), in each Layer and in each Component (the lowest level). The usual scenario is that a setting will override from the Top Down – i.e. a setting at a high level overrides that setting at lower levels. Alternatively, it may be Default Down – i.e. setting at a high level establishes a default that can be overridden by a setting at a lower level.

Enable	Top down - Algorithm overrides Layer which overrides Component.
CoordinateSystem	Default down – Component overrides Layer which overrides Algorithm.
ValidCellByComponentRule	Default down – Layer overrides Algorithm.
UnderviewInterpolation	Top down - Algorithm overrides Layer which overrides Component.
InterpolationNearest	Top down - Algorithm overrides Layer which overrides Component.

Shadow	Top down - Algorithm overrides Component.
Highlight	Top down - Algorithm overrides Component.

10.2(a)

A shadow object is used to control the sun angle and intensity in an Intensity component. A highlight object is used to control the specular highlight (or what used to be called 'wet look') in an Intensity component. Shadow and highlight information defined in the algorithm override any settings defined for the Intensity component.

The Scale factor can be used to increase or decrease the depth of the shadow and the intensity of the highlight. Both of these calculations rely on statistics for the mean value of the change from one cell to the next. These statistics are acquired from the raster data being rendered (and in this case the CellToCellMean ought to be set to -1). If this value is not appropriate for your data, you can set it manually.

10.4(a)

In most cases, color tables will be loaded from external files in any of the numerous supported file formats. The file is likely to be located in the ColorTables directory in the MapInfo Pro installation. However, it is easy to define your own color tables and by using interpolation you can often do so using very few actual color elements. For example, you can define a 256 color "Pseudocolor" table by defining two color elements – Blue with index 0 and Red with index 255 and use HSL color space interpolation to fill in all the colors between.

10.6(a)

The layer type determines what components the layer will use and the rendering algorithm employed. We currently support LUTColor, RGBColor and Image layer types. A LUTColor layer has a color component that transforms data to a color index into a color table. A RGBColor layer has red, green and blue color components and each component transforms data to a color index into a fixed color table. Image acquires color data directly from an existing imagery source.

All layers can be enhanced by sun shadow and highlighting in an Intensity component. All layers can implement opacity modulation by using an opacity component.

Note that each component can be connected to a different raster source and there are no restrictions placed on this. The raster sources can have different structures, have different cell size and have different coordinate systems.

10.6(b)

ColorIntensity is a balance control that allows you to vary the contribution of the color components and the intensity component. The value ranges from 0 to 100. A value of 0 gives you no color and full intensity. A value of 100 gives you full color and no intensity. Color ramps from 0 - 100% over the range 0 to 50, and stays at 100% from 50 to 100. Intensity is at 100% over the range 0 to 50 then ramps down to 0% from 50 to 100. A setting of 50 gives full color and intensity. A setting of 60 gives full color and 80% intensity.

10.7(a)

LUTColor can contain a Color, an Intensity and an Opacity component. RGBColor can contain a Red, a Green, a Blue, an Intensity and an Opacity component. Image can contain an Image, an Intensity and an Opacity component. There should only be one of each, although you can disable any of the components in a layer.

10.7(b)

When you establish a data transform range, this may cover a subset of the source data statistical range. That means the system has to assign a color to values that lie below the lower cut off and above the upper cut off. By default, the system assigns the lowest and highest color respectively. If you turn on ClipToRange then the system will assign the background color instead.