# Lunaserv Configuration Documentation

Last Revised 2016/01/25

# Lunaserv Configuration Documentation

## Introduction

All Lunaserv configuration is done in YAML format. YAML is a simple text-based format that is both human & computer readable. Configuration files must exist in under the Lunaserv configuration directory. Spatial reference system (SRS) config files go under the srs subdirectory, and layer config files go under the layer subdirectory.

When config files refer to colors, they are always written in hex as either 6 digit (rrggbb) or 8 digit (rrggbbaa). The alpha channel is specified as 00 being fully transparent and ff being fully opaque. RGB values work as with html colors.

In config files that access a database, filters may be defined. These filters can be geographic bins, or extra WMS parameters. In the case of WMS parameters, the filter may have a specified default. Take this section of the luna_day_night layer as an example

```
:point_query: "select ST_X(pg_solar) as longitude, ST_Y(pg_solar) as
latitude from lunar_subpt where time = date_trunc('minute',
#{TIME_CLAUSE});"
:filter_params:
- TIME
:TIME_CLAUSE: "#{TIME}::timestamp at time zone 'UTC'"
:TIME_CLAUSE_DEFAULT: "now()"
```

In the query, the snippet '#{TIME_CLAUSE}' is substituted with either the TIME_CLAUSE parameter or the TIME_CLAUSE_DEFAULT parameter. If an extra WMS parameter of TIME is specified in the request URL, that parameter will be SQL escaped and quoted and substituted for #{TIME} in the TIME_CLAUSE which, in turn, will be substituted in the query. If no TIME parameter is given in the request, TIME_CLAUSE_DEFAULT will be placed in the query.

The additional config option 'filter_params' is an array of filters to look for. Each item listed must have an associated clause entry. A default entry for each filter is optional, and if omitted, the clause will simply be blanked out in the query if no parameter is given in the request.

If the term #{BINS} appears in the query (such as the one in Sample Layer #5), Lunaserv will compute which 5° bins overlap the requested projected area, and those bin ids will be returned, comma separated, in the query where #{BINS} appears. This can be useful for large vector datasets by pre-computing which bins each shape overlaps, so a request can attempt to render the smallest subset of shapes possible to fulfill the request.

## Spatial Reference Systems

There are two types of SRS configs supported, generic and IAU2000. A generic SRS must have four parameters:

- human_name: The text that will be advertised in a capabilities request. If configuring a standard SRS, this should match the SRS name in the standard.
- srs_match: This is either a text string or a Ruby regular expression used to match against the SRS field of requests
- srs_proj: This is the proj4 string for this SRS. Substitutions referring to fields in the srs_match regular expression are allowed
- body: This is the planetary body this SRS applies to. If the projection is universal, this field is left blank.

### *Sample SRS #1: EPSG:4326*

While EPSG:4326 is technically an Earth-based SRS, because it deals just in latitude and longitude, it is useful for exchanging data with a variety of WMS clients for any planetary body. Therefore, we declare it as a universal SRS here.

```
---
:human_name: EPSG:4326
:srs_match: EPSG:4326
:srs_proj: +init=epsg:4326
:body:
```

### *Sample SRS #2: JMARS:1*

This is an example of a custom non-standard SRS. Because JMARS is used to support multiple planetary bodies, this is also specified as universal.

```
---
:human_name: JMARS:1,up_lon,up_lat
:srs_match: !ruby/regexp /^JMARS:1,([-+]?\d+?\.?\d*?),([-+]?\d+?\.?\d*?)
$/
:srs_proj: +proj=ob_tran +o_proj=eqc +o_lat_p=#{$2} +o_lon_p=0
+lon_0=#{180-$1.to_f} +R=57.2957795131
:body:
```

An IAU2000 SRS config file actually implies an entire set of SRS definitions. This type of SRS is also a four line config file, but in this case, the parameters are a little different:

- iau_code: This is the body code for the planetary body. This code comes from NAIF.
- a: This is the semimajor radius of the ellipsoid axis

- b: This is the semiminor radius of the ellipsoid axis
- body: This is the planetary body represented by this IAU2000 definition

### *Sample SRS #3: IAU2000 Mercury*

This config defines IAU2000 SRS set for Mercury.

```
---
:iau_code: 199
:a: 2439700
:b: 2439700
:body: mercury
```

With this one file, the following projections are declared with Mercury specific parameters:

- Mercury 2000
- Sinusoidal
- North Pole Stereographic
- South Pole Stereographic
- Mollweide
- Sinusoidal AUTO
- Orthographic AUTO
- Mollweide AUTO

## Pyramidal TIFFs

Both raster and stamp layer types use pyramidal TIFFs (PTIFFs). These are specially formatted TIFF files where each layer decreases in both the x and y dimensions by half, and each layer is tiled at a size of 256x256. The stamp files also have a mask PTIFF in the same form, but only 1 bit per pixel specifying the appropriate mask for the PTIFF. Each PTIFF may also have an extents file describing the data in geographic terms to Lunaserv, or the PTIFF may have proper GeoTIFF information embedded in the image. The extents file is 4 or 5 lines long. The first 4 lines are min x, max x, min y, max y; the units of those values should match the projection. In the case of simple cylindrical, the units are degrees. The 5th line is optional; if the PTIFF is projected in simple cylindrical, then it may be omitted, but if it is in some other projection, the 5th line is the appropriate proj4 string describing the projection.

There are multiple options to make a PTIFF. The Lunaserv team has made available pnm2ptif which is capable to transforming a PNM file to a ptif, optionally with GeoTIFF information embedded, and also optionally with a mask file. The open source vips software may also be used.

Pnm2ptif usage:

```
usage: pnm2ptif [options] in_file out_file
   or: pnm2ptif [options] out_file < in_file
   or: pnm2ptif --version

options:
-j, --jpeg          Use JPEG compression. Quality 0-100 optional, default
is 75.
-m, --maskfile      Write a mask PTIF (assumes DN 0 means NULL).
-p, --projection    Embed projection information using GeoTIFF tags.
                    This option should be supplemented with a proj4-
formatted projection string (example: "+proj=latlong"),
                    and the following four boundary-specification options
must be included in conjunction.
  -x, --min-x       Lower X bound in projection space.
  -X, --max-x       Upper X bound in projection space.
  -y, --min-y       Lower Y bound in projection space.
  -Y, --max-y       Upper Y bound in projection space.
-r, --restrict      Restrict the smallest image dimension, default is 1.
-s, --skipspecial   Skip special pixels for averaging purposes.
-v, --verbose       Display more information.
-V, --version       Print version.
-z, --zip           Use zip (deflate) compression. Level 1-9 optional,
default is 6.
```

Example vips command line:

```
vips im_vips2tiff "src" "dst.tif":jpeg:75,tile:256x256,pyramid
```

## Layers

Layer config files under in the lunaserv/layer directory are further divided by the planetary body they represent. For some layers, such as a lat/long grid, there is no body, so they are placed in a subdirectory called universal. If a layer requires additional files, such as basemap tiles, these are placed in a subdirectory matching the layer's name. The subdirectory can be overridden with a *dirname* option. Layers also have some configuration items, such as colors, that can be overridden using an optional parameter in the WMS request. The parameter to override an option is specified in the form of layer name, underscore, then option name. For example, if the user wanted a layer called universal_grid10 to be rendered in blue, UNIVERSAL_GRID10_STROKE_COLOR=0000ffff could be added to the WMS URL.

Several different layer types are supported:

- grid: Generic lat/long grids of varying sizes
- illumination: topography based dynamic illumination
- night: basic night-side shading using NAIF's SPICE toolkit to get the sun's position.
- numeric: A special VICAR based layer for transporting numeric data in a raster form
- raster: Raster basemaps
- shading: Provides a semi-transparent shaded hemisphere opposite a center point. This is most often used to implement a day/night type layer when SPICE is unavailable.
- stamp_db: small raster areas using a DB to find them
- stamp_file: small raster areas with a static set of files
- vector_db: Vector data originating in a PostgreSQL database
- vector_file: Vector data originating in a file formatted specifically for Lunaserv
- vector_shapefile: Vector data originating in a standard shapefile

Regardless of layer type, there are several parameters that must be specified:

- type: specifies which layer type this is
- wms_name: The body is prepended to this name to make the WMS layer name.
- title: A human readable name.
- abstract: A chunk of text describing the layer that will be presented in Lunaserv's capabilities.
- queryable: true/false Can this layer respond to feature requests? Currently only vector_db supports this.
- opaque: true/false Is this layer largely opaque? This would typically only be true for basemaps
- minx/miny/maxx/maxy: lat/long bbox for this layer
- keywords: An array of keywords to facilitate clients searching for this layer.
- public_layer: true/false Should this layer be offered without authentication. (default:false)

## *Grid Layers*

Grid layers create an automatic lat/long grid of a configurable color and spacing. Two parameters are added to the base options to configure a grid layer:

- spacing: The size of the grid in degrees

  *The following options are able to be overwritten:*

- stroke_color: an 8 digit hex value to draw the grid with
- fill_color: 8-digit hexadecimal string specifying RGBA values for closed-polygon fill color of vector layer (default: no fill / transparent)
- background: 8-digit hexadecimal string specifying RGBA values for background color of text annotations (default: no background / transparent)
- font_size: if not specified, defaults to a size derived from the output image size
- diameter: diameter of plotted single points in pixels
- padding: thickness of padding around annotations in pixels

### *Sample Layer #1: universal_grid10*

```
---
:type: grid
:wms_name: grid10
:public_layer: true
:title: 10 Degree Grid
:abstract: Dynamically generated
:queryable: false
:opaque: false
:minx: -180
:maxx: 180
:miny: -90
:maxy: 90
:keywords:
- grid
:stroke_color: "000000ff"
:spacing: 10
```

## *Illumination Layers*

Illumination layers dynamically generate topography based shadowing based on a DEM stored in an ISIS cub file. If multiple files are specified, it is assumed that they all cover the same area, but are in different resolutions. In this case, Lunaserv will use the lowest resolution that is higher than the requested resolution of a GetMap request. Five additional options are required for this layer type:

- basemaps: An array of one of more cub files
- elevation_variance_km: The difference in km between the highest and lowest points in the DEM

- radius_km: The average radius of the body in km. This must match the DEM if the DEM measures elevation as an offset from a sphere.
- units: The units of the DEM, either :meters or :kilometers
- data_form: Does the DEM specify full :radii or is it an :offset from a sphere.

Additionally, the colors representing dark & light can be specified; this layer type will blend between the two colors as light areas fade into dark.

- dark_color: RGBA value for completely dark areas
- light_color: RGBA value for fully illuminated areas
- shadow_color: An 8-digit hexadecimal RGBA color to use for shadowed points. Default is 000000ff (opaque black)
- photometric_function: Specify a photometric function to use. Specify 'help' here to get a list of available photometric functions and exit.
- solar_distance: The distance from the planetary body to the Sun, in kilometers. If not supplied, will default to one AU (appropriate for the Earth and the Moon).

### *Night Layers*

Night layers generate a fast & simple nighttime shading opposite of the sub-solar point on the surface of the planet. The sub-solar point is calculated using NAIF's SPICE toolkit. Two options are required for this layer type *and can be overridden*:

- color: RGB color for the night side
- opacity: opacity of the night side color

### *Numeric Layers*

Numeric layers are intended primarily for use with JMARS. They generate floating point numeric data from a set of one or more ISIS cub files. The output of this layer type is always VICAR. If multiple files are specified, it is assumed that they all cover the same area, but are in different resolutions. In this case, Lunaserv will use the lowest resolution that is higher than the requested resolution of a GetMap request. There are four options added to the base options to configure a numeric layer:

- basemaps: An array of one or more cub files
- pre_offset: an offset to be applied to each value before scaling (default:0)
- post_offset: an offset to be applied to each value after scaling (default:0)
- scaling_factor: A factor to scale each value with (default:1)

### *Sample Layer #2: luna_wac_dtm_numeric*

This layer specifies a numeric layer with a source cub in absolute meters that we wish to present in kilometers offset from a sphere.


```
---
:type: numeric
:wms_name: wac_dtm_numeric
:public_layer: false
:title: LROC WAC Color Shaded Relief - Numeric
:abstract: "Shaded relief images were created from the GLD100 by
illuminating the surface from a given Sun direction and elevation above
the horizon. To convey an absolute sense of height, the resulting shaded
relief grayscale pixels were painted with colors that represent the
elevation (relative to the mean lunar radius). The Color Shaded Relief
map is available at 128 ppd in the same tiled format as the GLD100 as
well as global files at lesser resolutions. All of the Color Shaded
Relief products are available with and without latitude and longitude
grids (10° increments)."
:queryable: false
:opaque: true
:minx: -180
:maxx: 180
:miny: -90
:maxy: 90
:keywords:
- luna
- numeric
- jmarsCategory:By Instrument//WAC
- jmarsCategory:By Type//Global
:scaling_factor: 0.001
:post_offset: -1737.4
:basemaps:
- dtm.cub
```


## *Raster Layers*

Raster layers are most often global basemaps in Lunaserv. They are always simple cylindrical in a pyramidal TIFF format, and can be tiled if the basemap is too large to fit in a TIFF. Each PTIFF basemap tile must have a file of the same name with the extension changed to .ext that contains four lines. These lines are the minimum longitude, maximum longitude, minimum latitude, and maximum latitude of the tile. There is one parameter added to the base options to configure a raster layer:

- basemaps: An array of PTIFF tiles in simple cylindrical format.

### *Sample Layer #3: earth_blue_marble*

```
---
:type: raster
:wms_name: blue_marble
:public_layer: true
:title: NASA's Blue marble next generation w/ topography and bathymetry
:abstract: "Blue Marble: Next Generation offers greater spatial detail of
the surface and spans a longer data collection period than the original.
The original Blue Marble was a composite of four months of MODIS
observations with a spatial resolution (level of detail) of 1 square
kilometer per pixel. Blue Marble: Next Generation offers a years worth of
monthly composites at a spatial resolution of 500 meters. These monthly
images reveal seasonal changes to the land surface: the green-up and
dying-back of vegetation in temperate regions such as North America and
Europe, dry and wet seasons in the tropics, and advancing and retreating
Northern Hemisphere snow cover. From a computer processing standpoint,
the major improvement is the development of a new technique for allowing
the computer to automatically recognize and remove cloud-contaminated or
otherwise bad data - a process that was previously done manually.
Blue Marble: Next Generation improves the techniques for turning
satellite data into digital images. Among the key improvements is greater
detail in areas that usually appear very dark to the satellite (because a
large amount of sunlight is being absorbed), for example in dense
tropical forests. The ability to create a digital image that provides
great detail in darker regions without washing out brighter regions, like
glaciers, snow-covered areas, and deserts is one of the great challenges
of visualizing satellite data. The new version also improves image
clarity, and gives highly reflective land surfaces, such as salt flats, a
more realistic appearance."
:queryable: false
:opaque: true
:minx: -180
:maxx: 180
:miny: -90
:maxy: 90
```

```
:keywords:
- earth
:basemaps:
- marble1.tif
- marble2.tif
- marble3.tif
- marble4.tif
- marble5.tif
- marble6.tif
- marble7.tif
- marble8.tif
```

## *Shading Layers*

Shading layers are useful for implementing a day/night type of layer. It could also be used to mask off a hemisphere for any other reason. This layer is database driven and requires seven parameters in addition to the base options. This layer will also accept query filters as described above.

- color: 6 digit hex color
- opacity: 2 digit hex opacity
- dbhost, dbuser, dbpassword, dbname: database connection parameters
- point_query: Any query that returns a column named longitude and a column named latitude. This point represents the center of the non-masked hemisphere.

### *Sample Layer #4: luna_day_night*

This layer will shade the night side of the Moon based on the current time, or the WMS client may request a specific time to render. As long as this time exists in the database, it will render the appropriate shading.

```
---
:type: shading
:wms_name: day_night
:public_layer: true
:title: Nighttime shading
:abstract: "This layer does basic shading for the hemisphere away from
the sub-solar point."
:queryable: false
:opaque: false
:minx: -180
```

```
:maxx: 180
:miny: -90
:maxy: 90
:keywords:
- illumination
:color: "000000"
:opacity: "70"
:dbhost: aaron.resnet
:dbuser: opsuser
:dbpassword: opsuser
:dbname: lroc
:point_query: "select ST_X(pg_solar) as longitude, ST_Y(pg_solar) as
latitude from lunar_subpt where time = date_trunc('minute',
#{TIME_CLAUSE});"
:filter_params:
- TIME
:TIME_CLAUSE: "#{TIME}::timestamp at time zone 'UTC'"
:TIME_CLAUSE_DEFAULT: "now()"
```

### *Stamp DB Layers*

The stamp file layer will render small areas of raster data based on a set of files. The files must meet the same requirements as the raster base layers, but they need not be global, and they can be overlapping. The files are treated in priority order in the case of overlap. Stamp files have one additional file which is a 1-bit ptif representing the mask of the data in the main ptif. Only masked areas are rendered. The files are located using a database. The query used must return a column called "stamp_file". This layer type responds to both GetMap and GetFeatureInfo requests, and optional filters can be applied as described above. There are seven additional options required for this layer type:

- dbhost, dbuser, dbpassword, dbname: database connection parameters
- stamp_query: The DB query used to find stamps. The ptif path and filename should be returned as a column named "stamp_file".
- feature_query: The DB query used to do feature requests. The points field should be returned as in the shape_query, any extra columns returned will be serialized and sent to the WMS client as the feature data. In addition to filters, #{LON}, #{LAT}, and #{RADIUS} will be substituted in and will represent the search area for the feature request. #{FEATURE_LIMIT} will also be substituted in and represents the requested limit for number of features returned in the WMS request. (default:no feature queries)
- feature_search_radius: How many destination image pixels radius should the DB search be.

## *Stamp File Layers:*

The stamp file layer will render small areas of raster data based on a set of files. The files must meet the same requirements as the raster base layers, but they need not be global, and they can be overlapping. The files are treated in priority order in the case of overlap. Stamp files have one additional file which is a 1-bit ptif representing the mask of the data in the main ptif. Only masked areas are rendered. Only one additional option is required for this layer type:

- stamps: an array of ptif stamps to render

## *Vector DB Layers*

The vector db layer will render polygons, polylines, or points out of a database. Feature requests are supported with this layer type, and filter parameters may be specified as described above. The filters are applied to both GetMap and GetFeatureInfo requests. Several parameters in addition to the base options are available in this layer *and may be overidden*:

- stroke_color: An 8 digit hex color
- fill_color: An 8 digit hex color (note: polygons that go over the edge of a requested projection will not be filled) (default:no fill)
- diameter: The diameter in pixels of points (default:8)
- limit: Stop rendering after this number of shapes are rendered (default: unlimited)
- units: :degrees/:radians The units that the source data is in.
- dbhost, dbuser, dbpassword, dbname: database connection parameters
- shape_query: The DB query used to load shapes. The point data should be returned as 'points', an additional column called 'type' should return 'open' or 'closed' for each shape to indicate if it is a polygon or a polyline. If the type field is omitted, the shapes will be assumed to be polygons.
- feature_query: The DB query used to do feature requests. The points field should be returned as in the shape_query, any extra columns returned will be serialized and sent to the WMS client as the feature data. In addition to filters, #{LON}, #{LAT}, and #{RADIUS} will be substituted in and will represent the search area for the feature request. #{FEATURE_LIMIT} will also be substituted in and represents the requested limit for number of features returned in the WMS request. (default:no feature queries)
- feature_search_radius: How many destination image pixels radius should the DB search be.

### *Sample Layer #5: luna_roi*

```
---
:type: vector_db
```

```
:wms_name: roi
:public_layer: false
:title: LROC Regions of Interest DB
:abstract: "This layer is generated out of the LROC DB. The source of
this data is the ACT REACT DB."
:queryable: true
:opaque: false
:minx: -180
:maxx: 180
:miny: -90
:maxy: 90
:keywords:
- LROC
- ROI
- jmarsCategory:By Type//Shapes
:stroke_color: "99eeee88"
:units: :radians
:dbhost: host
:dbuser: username
:dbpassword: password
:dbname: dbname
:shape_query: "select
replace(replace(replace(replace(coalesce(ps_path::text, ps_poly::text),
'{(', ''), ' , ', ','), '),(', ';'), ')}', '') as points, case when
ps_poly is not null then 'closed' else 'open' end as type from
target_suggestion join product_search_bin_target_suggestion using
(target_suggestion_id) where product_search_bin_ids &&
'{#{BINS}}'::integer[] and (ps_path is not null or ps_poly is not null);"
:feature_search_radius: 5
:feature_search_query: "select target_suggestion.*,
replace(replace(replace(replace(coalesce(ps_path::text, ps_poly::text),
'{(', ''), ' , ', ','), '),(', ';'), ')}', '') as points, case when
ps_poly is not null then 'closed' else 'open' end as type from
target_suggestion join product_search_bin_target_suggestion using
(target_suggestion_id) where product_search_bin_ids &&
'{#{BINS}}'::integer[] and ((ps_path is not null and ps_path && scircle
'<(#{LON}d, #{LAT}d), #{RADIUS}d>') or (ps_poly is not null and ps_poly
&& scircle '<(#{LON}d, #{LAT}d), #{RADIUS}d>')) limit #{FEATURE_LIMIT};"
```

## *Vector File Layers*

This layer type renders much like the vector db layer type, but the source of the data comes from flat files. The files can contain points, polygons, polylines, or annotations in the following formats:

- p:lon,lat a single point
- c:N:lon,lat;lon,lat… a polygon with N points.
- o:N:lon,lat;lon,lat… a polyline with N points.
- a:lon,lat:label an annotation at the given point.

There are nine parameters for this layer in addition to the base options *and can be overridden*:

- stroke_color: An 8 digit hex color
- fill_color: An 8 digit hex color (default:no fill)
- background: a 6 digit hex color to draw behind annotation text (default:no background)
- units: :degrees/:radians What units is the data in?
- font_size: The font size to render annotations with.
- padding: How much empty space should be left around each annotation.
- file: The file to load the shape data from.
- limit: Stop rendering after this many shapes.
- diameter: diameter of plotted single points in pixels
- point_color: 8-digit hexadecimal string specifying RGBA values for point color (defaults to stroke color if not specified)

### *Sample Layer #6: universal_overview_annotation*

```
---
:type: vector_file
:wms_name: overview_annotation
:public_layer: true
:title: Overview Annotation
:abstract: "This layer shows a few select labels to assist with location
and orientation."
:queryable: false
:opaque: false
:minx: -180
:maxx: 180
```

```
:miny: -90
:maxy: 90
:keywords:
- annotation
:units: :degrees
:font_size: 16
:stroke_color: "64ff64ff"
:background: "000000ff"
:antialiasing: true
:padding: 10
:file: labels
```

## *Vector Shapefile Layers*

This layer type renders shapefiles. Most types of shapefile data are supported. Four parameters are available for this layer type in addition to the base options *and can be overridden*:

- stroke_color: An 8 digit hex color
- fill_color: An 8 digit hex color (default: no fill)
- diameter: The size, in pixels, to render points (default:8)
- shapefile: The shapefile to render.
- font_size: font size; if not specified, defaults to a size derived from the output image size.
- padding: Minimum padding, in pixels, between text labels.  Default is 10.
- background: 8-digit hexadecimal string specifying RGBA values for text background color of shapefile layer (default: no background / transparent)
- label_template:  Formatted string for labeling shapes, with DBF field names inserted as follows: "Example text, including #{DB_FIELD_1} and also #{DB_FIELD_2}."

### *Sample Layer #7: earth_world_border*

```
---
:type: vector_shapefile
:wms_name: world_border
:public_layer: true
:title: World Borders
:abstract: "Provided by Bjorn Sandvik, thematicmapping.org

If you have any comments about this dataset, please post them on this
page.
```

```
Use this dataset with care, as several of the borders are disputed.

The original shapefile (world_borders.zip, 3.2 MB) was downloaded from
the Mapping Hacks website: http://www.mappinghacks.com/data/

The dataset is available under a Creative Commons Attribution-Share Alike
License. If you use this dataset, please provide a link to this website."
:queryable: false
:opaque: false
:minx: -180
:maxx: 180
:miny: -90
:maxy: 90
:keywords:
- earth
:stroke_color: "ffff00ff"
:shapefile: TM_WORLD_BORDERS-0.3.shp
```

## Web interface configuration

Information about the WMS server(s) and available layers/parameters is provided to the web interface via a YAML configuration file stored within the web server's directory structure as public/config/lunaserv.yml. This file stores a collection of information with a primarily hierarchical structure, each item of which is described here.

### *WMS_URLS*

This entry, separate from the rest of the configuration file, denotes a list (each item beginning with '-' characters as per YAML syntax specification in general) of URLs through which WMS can be directly accessed to make requests. Only the base URL (example: https://my.wms.server/) is needed, as the rest of the URL will be constructed automatically for each request. Only one URL needs to be specified, but multiple URLs can be beneficial. Requests for sections of imagery are evenly spread across all provided URLs.

### *DEFAULT_BODY*

This entry simply specifies the name of the body (matching its NAME property) to be loaded at startup. Bodies and their properties are discussed in more detail below.

## *UNIVERSAL_LAYERS*

This is a special section for layers which should be provided no matter which planetary body is selected and whose parameters on the server are fixed. A good example is a latitude/longitude grid. This section is syntactically exactly like the body-dependent LAYERS sections, discussed in more detail below.

## *BODIES*

This section is the bulk of lunaserv.yml; it specifies the bodies that are to be provided in the interface, and the layers associated with each body. The interface provides a dropdown list control in the "Map Options" dialog box which can be used to select from the available bodies. If only one body is specified in this section, the interface will respond appropriately by omitting this dropdown control entirely. Within BODIES is a hierarchical structure of information arranged as follows, where angle brackets indicate variable text that is explained below, and non-bracketed text is literally as shown:

```
BODIES:
 <BODY_ID>:
  NAME: <Textual body name>
  BODY_CODE: !!str <IAU_PLANETARY_CODE>
  RADIUS: <RADIUS>
  DEFAULT_BASEMAP: <default_basemap_name>
  LAYERS:
   <ELEMENT_ID>:
    NAME: <Textual element name>
    TYPE: <ELEMENT_TYPE>
    RADIO_GROUP: <RADIO_GROUP_NAME>
    SERVER_LAYERS: <layers_on_server>
    QUERIABLE: <QUERIABLE>
    ADDITIONAL_WMS:
     <WMS_PARAM_NAME>: <WMS_PARAM_VALUE>
    <SUB_ELEMENT_ID>:
     <...>
```

Following are explanations of the variable text:

Within the BODIES entry, many bodies may be present; there should be a chunk as above for each one, all on the same level, headed up by a <BODY_ID> and one level deep from the BODIES label.

<BODY_ID>: This is a unique identifier for the body; it will not be visible in the interface, but it should be unique for each body, and descriptive of the body it denotes. Example: MOON.

<Textual body name>: This is the name of the planetary body as it will be presented as a choice for the user. Example: “The Moon”.

<IAU_PLANETARY_CODE>: This is the section of the IAU2000 projection specifier which denotes the body. For bodies in our Solar System, this code begins with a digit identifying the planet and then concludes with either “99” to specify the planet itself or “01”, “02” etc. to specify a moon. For example, Earth's Moon is “301” and Mars is “499”. Note that we cast as a string with “!!str”, rather than leave it as a number; this is because the code is used in various places internal to the Lunaserv interface as character data rather than numerical data.

<RADIUS>: The radius of the planetary body, expressed in meters.

<default_basemap_name>: The name on the WMS server(s) of the layer to be used as the initially-visible basemap when this body is selected.

Each body should have a LAYERS section, which contains all the layers that should be available in the interface, and can be customized to include menus, submenus, radio button groups (of which only one layer may be selected at a time), and checkboxes (for layers that may be turned on and off independently of anything else). Within the layers section, an <ELEMENT_ID> and its contents each define one element of the overall layer selection menu, and very few or very many can be defined.

<ELEMENT_ID>: As with <BODY_ID>, this is a unique identifier for the element in the menu. It will not be visible in the interface, but should be a unique (and sensible) descriptor for the element. Example: BASEMAPS.

<Textual element name>: This is the label that will be used for the element as presented to the user. Example: “Basemap imagery”.

<ELEMENT_TYPE>: This can take one of several values, and specifies the type of element in the menu. Currently, three types are available:

- MENU: An expandable/collapsible menu or submenu of choices. These may be nested indefinitely.
- RADIO: A layer that is accessed via radio button, a control which, when selected, automatically deselects all other options in the same RADIO_GROUP (described below). This is useful for a set of layers which are meant to be viewed on an “either this one or that one, or neither, but never both” basis, rather than simultaneously.
- CHECKBOX: A layer that is accessed via checkbox and can be turned on and off independently of other selections.

<RADIO_GROUP_NAME>: This property only applies to elements of type RADIO, and is an identifier which classifies a layer choice within a set of mutually exclusive choices. You can have as few or as many radio groups as you want, and a radio group may be divided across multiple submenus (and one submenu may even contain radio button options from multiple radio groups, if desired). Basemaps should all be in a radio group together, and the special value BASE should be used as the RADIO_GROUP_NAME for the basemaps.

<layers_on_server>: This property applies only to layer elements (RADIO or CHECKBOX). Here, the names of the associated layers as they are called on the WMS server(s) should be specified. More than one WMS layer may be associated with a single selection in the interface; to do this, separate the names of the WMS layers with commas. In this situation, enabling this selection in the interface will cause all of these layers to be requested, and disabling this selection will turn them all back off.

<QUERIABLE>: This property only applies to layer elements (RADIO or CHECKBOX), and should have a value of YES if this layer selection is queriable on the server for feature information. The QUERIABLE entry can be omitted, and will be assumed NO if so. As of right now, the result of querying for features has been designed specifically with LROC's needs in mind. The function show_info(e) in public/javascripts/lunaserv.js converts a response from the server into presentable data, packages it into the lunaserv_thumbnail_container dialog box present in the HTML, and sets that dialog box visible to the user. This code can be modified if a different reaction to the feature query is desired.

The ADDITIONAL_WMS entry is optional and only applies to layer elements (RADIO or CHECKBOX). Under ADDITIONAL_WMS, any number of arbitrary parameters to be sent to the WMS server(s) along with this layer can be supplied, in the form NAME: VALUE.

<SUB_ELEMENT_ID>: This entry only applies within elements of type MENU, and as few or as many can be specified as desired. This is an ID which, just like ELEMENT_ID, will not be visible in the interface to the user, but must be unique and should be descriptive of the element it denotes, for configuration file readability. Within any section headed by a <SUB_ELEMENT_ID>, all the same entries from within the <ELEMENT_ID>-headed section are available, and from the given properties a sub-element will be constructed and nested into the menu containing the sub-element.