# Entity-specific Properties

## File VIEW

Defines a View entity. Views store a pre-configured way of viewing multiple datasets. This type of entity will have these additional properties:

| | |
|---|---|
| view_type | Specifies the type of view. A view that contains children, should have its "view_type" property set to "container", and should include a property named "entity_container" which lists the view children.<br><br>If the view is not a container, its "view_type" must be set to the ID of a View Program entity. Please review the View Programs section for more information on these programs. |
| entity_container | Used only if the "view_type" property is set to "container". This property contains the list of children views for the view entity. This property is made by concatenating the IDs of the children views separated by spaces. |
| input_value_#id# | Used only for views that use a View Program. Contains the value for the input. Replace the #id# placeholder by the actual id used by the input.<br><br>Depending on the value specified for "input_type_#id#" (see types below in this table) the value will have a different interpretation:<br><br>0 – A numeric value, can have decimals<br>1 – A date-time, a 64bit integer value representing a timestamp. Typically a Linux epoch time.<br>2 – An alphanumeric value<br>3 – An identifier for an entity<br>4 – A list of attributes, separated by spaces<br>5 – Ignored, attribute query properties are listed below in this table.<br>6 – A Boolean value, "0" for False and "1" for True.<br>7 – Ignored, color legend properties are listed below in this table.<br>8 – Ignored, drill hole properties are listed below in this table. |
| input_label_#id# | Used only for views that use a View Program. Contains the label for the input. Replace the #id# placeholder by the actual id used by the input. |

| | |
|---|---|
| input_type_#id# | Used only for views that use a View Program. Specifies type of the input. Replace the #id# placeholder by the actual id used by the input. The possible input types are:<br>0 – Numeric<br>1 – Date<br>2 – String<br>3 – Entity<br>4 – Attribute Set<br>5 – Query Expression<br>6 – Boolean (Checkbox)<br>7 – Color Legend<br>8 – Drill-hole Settings |
| input_filter_#id# | Used only for views that use a View Program. Specifies the type of the input. Replace the #id# placeholder by the actual id used by the input. Must be non-zero for type 3 (Entity), contains a numeric value containing a binary mask identifying which types of entities can be potentially selected. Use binary OR to merge different types into a single mask. The values identifying each identity type are (in hexadecimal):<br>0x01 – Voxel Terrain<br>0x02 – Voxel Block Model<br>0x04 – Voxel Generator<br>0x08 – Realtime Voxel Terrain<br>0x10 – Indexed Point Cloud |
| color_legend_mode | Used only when an input of type "Color Legend" is included in the object. Defines which color model should be applied to the object view. Can have one of these two values:<br>1. The view will use a single color to the entire object<br>2. The will will apply a gradient based on data values |
| color_legend_static_r | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "1". This property contains the Red component in the static color. This property ranges from 0 to 255. |
| color_legend_static_g | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "1". This property contains the Green component in the static color. This property ranges from 0 to 255. |
| color_legend_static_b | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "1". This property contains the Blue component in the static color. This property ranges from 0 to 255. |
| color_legend_attribute | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "2". This property contains the identifier of the attribute that will be used to derive the color for the object. |

| | |
|---|---|
| color_legend_attribute_index | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "2". This property contains the index of the attribute that will be used to derive the color for the object. This index is respective to the full list of available attributes in the object. |
| color_legend_range_min | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "2". This property contains the minimum attribute value to be mapped to the lower end of the color gradient. |
| color_legend_range_max | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "2". This property contains the maximum attribute value to be mapped to the higher end of the color gradient. |
| color_legend_range_step | Used only when an input of type "Color Legend" is included in the object and "color_legend_mode" is set to "2". This property defines the magnitude of each step in the attribute's space that will mapped as a single color in the the color gradient. |
| attribute_query_mode | Used only when an input of type "Query Expression" is included in the object. This property defines which attribute filter mode will be used. There are two possible modes:<br>1. Use attribute query builder<br>2. Use attribute query string |
| attribute_query_string | Used only when an input of type "Query Expression" is included in the object and the "attribute_query_mode" is set to "1". This property contains a query string. |
| attribute_query_builder | Used only when an input of type "Query Expression" is included in the object and the "attribute_query_mode" is set to "1". This property contains a list of space-separated terms. Each term contains four items separated by commas. Each of these items corresponds to:<br>1. A string denoting the attribute identifier<br>2. Minimum value for the attribute (inclusive)<br>3. Maximum value for the attribute (exclusive)<br>4. Boolean operator to concatenate the expression following in the next term. Use "&" to perform a boolean AND and use "\|" to perform a boolean OR. |
| drillholes_show_collars | Used only when an input of type "Drill-hole Settings" is included in the object. Set to "1" to make the drill hole collar geometry appear in the view. Set to "0" to hide drill hole collars for the object. |
| drillholes_show_collar_labels | Used only when an input of type "Drill-hole Settings" is included in the object. Set to "1" to make the drill hole collar labels appear in the view. Set to "0" to hide drill hole labels for the object. |

| | |
|---|---|
| drillholes_show_cylinders | Used only when an input of type "Drill-hole Settings" is included in the object. Set to "1" to make the drill hole cylinder geometry appear in the view. Set to "0" to hide drill hole cylinders for the object. |
| drillholes_show_attributes | Used only when an input of type "Drill-hole Settings" is included in the object. Set to "1" to make the drill hole attribute labels appear in the view. Set to "0" to hide drill hole attribute labels for the object. |
| drillholes_attributes | Used only when an input of type "Drill-hole Settings" is included in the object. Set to "1" to make the drill hole collar geometry appear in the view. Set to "0" to hide drill hole collars for the object. |
| tube_legend_attribute | Specifies the attribute ID for the attribute that will be used to compute cylinder diameter. |
| tube_legend_radius | Used when "tube_legend_scale_mode" equals "1", this defines the radius of all cylinders. |
| tube_legend_radius_max | Used when "tube_legend_scale_mode" equals "2", this defines the maximum radius of cylinders. |
| tube_legend_radius_min | Used when "tube_legend_scale_mode" equals "2", this defines the minimum radius of cylinders. |
| tube_legend_range_max | Used when "tube_legend_scale_mode" equals "2", this defines the maximum value for the attribute that will be mapped to the maximum cylinder radius. |
| tube_legend_range_min | Used when "tube_legend_scale_mode" equals "2", this defines the minimum value for the attribute that will be mapped to the minimum cylinder radius. |
| tube_legend_range_step | Used when "tube_legend_scale_mode" equals "2", this defines the discrete step increases in attribute value. |
| tube_legend_scale_mode | Set to "1" for constant cylinder radius. Set to "2" for attribute-dependent cylinder radius. |

## File VOXSURF

Defines a Voxel Terrain Surface. This type of entity will have these additional properties:

| | |
|---|---|
| source | Contains the ID of the entity that was used to produce the Voxel Terrain Surface |
| source_type | Contains the type of the entity that was used to produce the Voxel Terrain Surface. Can be either “RAWPC” (raw point cloud) or “HM” (heightmaps). |

| | |
|---|---|
| source_ortho | Contains the ID of the raw orthoimagery set used for the Voxel Terrain Surface. |
| normals | Contains “1” when the Voxel Terrain Surface contains normal maps, “0” otherwise. |

## File IDXORTHO

Defines an indexed ortho-imagery dataset. This type of entity will have these additional properties:

| | |
|---|---|
| source | Contains the ID of the ORTHO entity that was used to produce the imagery dataset |
| image_format | Specifies which image formats should be included in the processed imagery dataset. It is possible to specify multiple targets. Target can be identify by the following values, using space as a separator between targets:<br><br>• DXT – Hardware compressed textures for Desktop and Web Browsers.<br>• ETC - Hardware compressed textures for Mobile<br>• BGRA - Raw, uncompressed imagery<br>• JPG - Software compressed textures using JPEG compression |

## File RAWBM

Defines a raw Block Model CSV file. This type of entity will have these additional properties:

| | |
|---|---|
| column_meta | This property is optional. Please look into the following section to see how to provide information about the columns in the raw Block Model:<br><br>Extended Block Model Metadata<br><br>Defines how to interpret and process each column (attribute) found in the Block Model's CSV file. This value must be the Base64 encoding of a single string composed of multiple comma-separated headers. There should be one of these headers for each column in the CVS file. Each header contains four terms separated by spaces.<br><br>The first term is a unique identifier for the column in the model. It must not contain any spaces nor commas.<br><br>The second term identifies the type of column, it can be one of the following:<br>• SKIP – The column will be ignored<br>• VALUE – The column contains a numeric value of unknown type<br>• UNIT – The column contains a numeric value that represents a quantity, like tonnage<br>• GRADE – The column contains a numeric value that represents a fraction of bigger whole, e.g. a percentage, grade or parts-per-million.<br>• INT – The column contains a signed integer value<br>• UINT – The column contains an unsigned integer value<br>• ENUM – The column contains values belonging to a discrete set of values, which can be alphanumeric if needed<br>• XC – X coordinate for block centroid<br>• YC – Y coordinate for block centroid<br>• ZC – Z coordinate for block centroid<br>• XL – X coordinate for block origin<br>• YL – Y coordinate for block origin<br>• ZL – Z coordinate for block origin<br>• XS – Block dimension along the X axis<br>• YS – Block dimension along the Y axis<br>• ZS – Block dimension along the Z axis<br><br>The third term specifies the Level of Detail operation that will be performed for the column. These are the possible values:<br>• AVERAGE – Average the values<br>• ADD – Will add values<br>• MIN – Preserve the smallest absolute values<br>• MAX – Preserve the largest absolute values<br>• MINORITY – Will preserve the least occurring value<br>• MAJORITY – Will preserve the most occurring value<br>• MULTIPLY – Will preserve the smallest volume-weighted values<br>• WEIGHTED(COL_ID) – Performs weighted average using a secondary column, specified by COL_ID. |

| | |
|---|---|
| block_model_type | Optional property to specify the type of block model. The possible values are:<br><br>"octree" - Denotes an octree-based block model. This is the default value.<br>"varz" - Denotes a variable-z octree (DataMine) |

## File VOXBM

Defines a Voxel Block Model. This type of entity will have these additional properties:

| | |
|---|---|
| source | Contains the ID of the raw Block Model data that was used to process the Voxel Block Model |
| translate_x | Number of units to translate the block model along the X direction. |
| translate_y | Number of units to translate the block model along the Y direction. |
| translate_z | Number of units to translate the block model along the Z direction. |
| scale_x | A proportion (0..1) to scale the block model along the X direction. |
| scale_y | A proportion (0..1) to scale the block model along the Y direction. |
| scale_z | A proportion (0..1) to scale the block model along the Z direction. |
| rotate_x | Euler angle rotation for X axis. |
| rotate_y | Euler angle rotation for Y axis. |
| rotate_z | Euler angle rotation for Z axis. |
| rotation_order | Rotation order. Must be one of: ZXZ, XYX, YZY, ZYZ, XZX or YXY |

## File VOXMT

Defines a Material Tracking Layer. This type of entity will have these additional properties:

| | |
|---|---|
| attributes | Contains a list of comma separated attributes that will be tracked by the layer. |
| tags | Contains a list of comma separated tags that will be tracked by the layer. Unlike attributes, tags are not originally included in the resource model attribute schema. |
| resource_model | Contains a Base64 string that defines a list of resource models, allowing to define a timestamp and a series of tags for each model .<br><br>The models are listed as comma-separated terms, where each term corresponds to one resource model. Each term is a list of space-separated entries, following this sequence:<br><br><ENTITY_ID> <timestamp> [tag0=value] [tag1=value] ...<br><br>The first two terms are required:<br><br>• ENTITY_ID holds the unique identifier for the resource model entity<br>• timestamp is the time value at which the resource model was included in the tracking system<br><br>The following terms contain custom tag values, in the form:<br><br>tagId=value<br><br>The tags specified here should belong to the same set of tags that is specified for the "tags" attribute. Not all tags are required. |
| terrain_model | Contains the identifier for a Terrain Model entity. This terrain model will be used as a starting point for depletion of the resource models |
| resource_models_initialized | Set to zero to signal the resource models require re-processing. The system will set this to one after processing is complete. |
| | |

## File IDXPC

Defines an Indexed Point Cloud. This type of entity will have these additional properties:

| | |
|---|---|
| source | Contains the ID of the raw Point Cloud data that was used to process the Indexed Point Cloud. |

## File VOXPC

Defines an voxelized point cloud. This type of entity will have these additional properties:

| | |
|---|---|
| translate_x | Number of units to translate the point cloud along the X direction. |
| translate_y | Number of units to translate the point cloud along the Y direction. |
| translate_z | Number of units to translate the point cloud along the Z direction. |
| scale_x | A proportion (0..1) to scale the point cloud along the X direction. |
| scale_y | A proportion (0..1) to scale the point cloud along the Y direction. |
| scale_z | A proportion (0..1) to scale the point cloud along the Z direction. |
| rotate_x | Euler angle rotation for X axis. |
| rotate_y | Euler angle rotation for Y axis. |
| rotate_z | Euler angle rotation for Z axis. |
| rotation_order | Rotation order. Must be one of: ZXZ, XYX, YZY, ZYZ, XZX or YXY |

## File IDXMESH

Defines an indexed mesh. This type of entity will have these additional properties:

| | |
|---|---|
| source | Contains the ID of the raw Block Model data that was used to process the Voxel Block Model. |

| | |
|---|---|
| translate_x | Number of units to translate the mesh along the X direction. |
| translate_y | Number of units to translate the mesh along the Y direction. |
| translate_z | Number of units to translate the mesh along the Z direction. |
| scale_x | A proportion (0..1) to scale the mesh along the X direction. |
| scale_y | A proportion (0..1) to scale the mesh along the Y direction. |
| scale_z | A proportion (0..1) to scale the mesh along the Z direction. |
| rotate_x | Euler angle rotation for X axis. |
| rotate_y | Euler angle rotation for Y axis. |
| rotate_z | Euler angle rotation for Z axis. |
| rotation_order | Rotation order. Must be one of: ZXZ, XYX, YZY, ZYZ, XZX or YXY |

## File VOXMESH

Defines a voxelized set of meshes. This type of entity will have these additional properties:

| | |
|---|---|
| source | Contains the ID of the raw mesh entity used to process the Voxelized Mesh. |

| | |
|---|---|
| metadata | Contains a sequence of comma-separated terms. Each term describes an attribute for the voxelized set, and it is made from three space-separated items. These items are, in listing order:<br><br>1. Attribute ID - This identifies the attribute in the dataset<br>2. Attribute Type - Defines which type of attribute to use. Can be set to UID or VALUE.<br>3. Attribute Default Value - Default value for the attribute. Leave empty for UID type.<br><br>The set must have one attribute defined as UID, which will allow to link individual meshes in the set to their attributes in the property file.<br><br>See the Extended Voxelized Mesh Metadata topic for more information on how metadata for Voxelized Mesh Sets is defined. |
| translate_x | Number of units to translate the mesh along the X direction. |
| translate_y | Number of units to translate the mesh along the Y direction. |
| translate_z | Number of units to translate the mesh along the Z direction. |
| scale_x | A proportion (0..1) to scale the mesh along the X direction. |
| scale_y | A proportion (0..1) to scale the mesh along the Y direction. |
| scale_z | A proportion (0..1) to scale the mesh along the Z direction. |
| rotate_x | Euler angle rotation for X axis. |
| rotate_y | Euler angle rotation for Y axis. |
| rotate_z | Euler angle rotation for Z axis. |
| rotation_order | Rotation order. Must be one of: ZXZ, XYX, YZY, ZYZ, XZX or YXY |

## File RAWHM

Defines a set of orthoimagery that describes elevation. This type of entity will have these additional properties:

| | |
|---|---|
| elevation_min | Contains the minimum elevation value for the set. This value maps to a black pixel. |

| | |
|---|---|
| elevation_max | Contains the maximum elevation value for the set. This value maps to a white pixel. |

## File RAWDH

Defines source data for a Drill Hole entity. While this entity does not have any special properties, the entity must include a process.meta file. This is covered in the Extended Drill Hole Metadata section.

## File PROGRAM

Defines a voxel generator program. This type of entity will have these additional properties:

| | |
|---|---|
| code_origin | Contains the type of code used by the program. Currently, two types are supported:<br>• "PYTHON" - The code is a Python program<br>• "NETASM" - The code is provided as a compiled .Net assembly |
| code | If code_origin is "PYTHON", this property contains the Python code for the program, in Base64 format.<br>If code_origin is "NETASM", this property identifies the entry-point class for the Lambda program in the form: <FullyNamespacedClassName>:<DLLName> |
| program_type | Contains the type of the program. The program type can be one of the following:<br>• “VOXEL” – The program is for a voxel entity generator. This type only accepts "PYTHON" as code_origin.<br>• “VIEW” – The program creates a view This type only accepts "PYTHON" as code_origin.<br>• “REPORT” – The program creates a report. Reports can be have "PYTHON" or "NETASM" as code_origin |

## File VOXGEN

Defines a spatial entity that is the output of a voxel generator program. This type of entity will have these additional properties:

| | |
|---|---|
| program | Contains the ID of the voxel generator program that will be evaluated. |
| input_value_#id# | Contains the value for the input. Replace the #id# placeholder by the actual id used by the input. |
| input_label_#id# | Contains the label for the input. Replace the #id# placeholder by the actual id used by the input. |
| input_type_#id# | Specifies type of the input. Replace the #id# placeholder by the actual id used by the input. The possible input types are:<br>0 – Numeric<br>1 – Date<br>2 – String<br>3 – Entity<br>4 – Attribute Set<br>5 – Query Expression |
| input_filter_#id# | For Specifies type of the input. Replace the #id# placeholder by the actual id used by the input.<br>Must be non-zero for type 3 (Entity), contains a numeric value containing a binary mask identifying which types of entities can be potentially selected. Use binary OR to merge different types into a single mask. The values identifying each identity type are (in hexadecimal):<br>0x01 – Voxel Terrain<br>0x02 – Voxel Block Model<br>0x04 – Voxel Generator<br>0x08 – Realtime Voxel Terrain<br>0x10 – Indexed Point Cloud |

## File REPORT

Defines a report entity. A report entity takes a Program and runs it over a specified Region. The program gets to inspect the spatial data int the region and it can compute useful information by combining the different spatial attributes using custom logic.

Creating a report entity is not enough to produce the desired report files. After the entity has been created, it is also necessary to trigger its processing by a second REST call. Please visit the “Triggering reports” section to see how report jobs can be triggered.

This type of entity will have these additional properties:

| | |
|---|---|
| program | Contains the ID of the report program that will be evaluated. |

| | |
|---|---|
| lod | Specifies at which Level of Detail the report will run. |
| input_value_#id# | Contains the value for the input. Replace the #id# placeholder by the actual id used by the input. |
| input_label_#id# | Contains the label for the input. Replace the #id# placeholder by the actual id used by the input. |
| input_type_#id# | Specifies type of the input. Replace the #id# placeholder by the actual id used by the input. The possible input types are:<br>0 – Numeric<br>1 – Date<br>2 – String<br>3 – Entity<br>4 – Attribute Set<br>5 – Query Expression |
| input_filter_#id# | For Specifies type of the input. Replace the #id# placeholder by the actual id used by the input.<br>Must be non-zero for type 3 (Entity), contains a numeric value containing a binary mask identifying which types of entities can be potentially selected. Use binary OR to merge different types into a single mask. The values identifying each identity type are (in hexadecimal):<br>0x01 – Voxel Terrain<br>0x02 – Voxel Block Model<br>0x04 – Voxel Generator<br>0x08 – Realtime Voxel Terrain<br>0x10 – Indexed Point Cloud |

## File EXPORT

Defines an export entity. Export entities contain files in standard formats that were produced out of the spatial data in the project. For instance, a user may desire to export a mashup of terrain surfaces as a single terrain mesh. In this case, there would be one export entity that stores the mesh as an internal file.

Creating an export entity is not enough to produce the desired files. After the entity has been created, it is also necessary to trigger its processing by a second REST call. Please visit the “Triggering jobs” section to see how export jobs can be triggered.

This type of entity will have these additional properties:

| | |
|---|---|
| entity | Contains the ID of the entity that will be used as a source for the export operation |
| lod | Specifies at which Level of Detail the report will run. |

| | |
|---|---|
| region | A region can be expressed in the REST interface as a single string, containing the following format:<br><br>1,<min_height>,<max_height>,<polygon_point0_x>, <polygon_point0_y>, <polygon_point1_x>, <polygon_point1_y>, ...<br><br>The format is a list of numbers separated by commas. The first number must be 1, this identifies this is an extruded polygon region. The second two numbers are world coordinates for the vertical bounds of the region. The following numbers are the X, Y coordinates of each region point. The polygon points are defined in counter-clockwise order.<br>For example, the following region string defines a 10-unit cube centered in the project's origin:<br>1,-5,5,-5, 5,5,5,5,-5,-5,-5 |
| export_type | Specifies the type of data that will be exported. Can be one of these:<br>• mesh – Exports the surface mesh for the entity (.OBJ format)<br>• points – Exports a point cloud (CVS format)<br>• ortho – Exports ortho-imagery (TIF+TFW) |

Export entities have one special file attachment. This file is named "export.zip" and contains a ZIP archive of the exported data.

## Region entity

Entities of type "REGION" describe a 3D scope where export jobs, reports and other tasks will be performed.

This type of entity will have these additional properties:

| | |
|---|---|
| region | A region can be expressed in the REST interface as a single string, containing the following format:<br><br>1,<min_height>,<max_height>,<polygon_point0_x>, <polygon_point0_y>, <polygon_point1_x>, <polygon_point1_y>, ...<br><br>The format is a list of numbers separated by commas. The first number must be 1, this identifies this is an extruded polygon region. The second two numbers are world coordinates for the vertical bounds of the region. The following numbers are the X, Y coordinates of each region point. The polygon points are defined in counter-clockwise order.<br>For example, the following region string defines a 10-unit cube centered in the project’s origin:<br>1,-5,5,-5, 5,5,5,5,-5,-5,-5 |

Revision #3
Created 17 March 2025 14:51:42 by admin
Updated 17 March 2025 21:07:12 by admin