ESRI
EXTERNAL

Oriented Imagery Catalog Schema

User Documentation | August 9th, 2022

380 New York Street
Redlands, California 92373-8100 USA
909 793 2853
info@esri.com
esri.com

esri | THE SCIENCE OF WHERE™

This document applies to Oriented Imagery 1.1.2. Oriented Imagery and associated tools are currently provided as a prototype and for testing only. The functionality has not been exhaustively tested and is not currently covered under ArcGIS support. Please address questions or suggestions related to this workflow to Esri Community or to ImageManagementWorkflows@esri.com.

Contents

Overview

We refer to "oriented" imagery as nontraditional mapping imagery. Such imagery may include street-side imagery taken from a mobile platform, close-range inspection images (e.g., transmission towers, bridge repairs) captured using drones or underwater rovers, or smartphone pictures taken by workers in the field. It also includes many forms of oblique imagery for viewing the different sides of buildings—in property assessment or for the inspection of power lines, for example. Such imagery is not typically suitable or desired for orthorectification due to a number of factors, including high oblique orientation (camera field of view partially or completely above the horizon), insufficient metadata, confusing content in the background, and/or subject matter with significant vertical structure. Although the ArcGIS mosaic dataset has robust support for nadir and low oblique imagery, the data model is not suitable for many forms of oriented imagery.

This document defines a schema for describing an oriented imagery catalog (OIC) for use in the ArcGIS platform. The OIC is defined as a JSON that references a point-based feature service that defines the camera location and orientation as well as metadata. The design should allow for scaling to many millions of images in a single catalog.

The OIC is used to find appropriate imagery for any location and enable an indication of the image coverage. The imagery should then be displayed, and if the accuracy of the orientation is sufficient, provide a mapping between ground features and image features as well as the ability to take measurements, superimpose the display of features into the images, or annotate and record features in the image with the location in map space also being recorded.

The range of applications includes 2D web maps, 3D scenes, mobile apps, and desktop applications such as ArcGIS Pro.

Oriented Imagery is primarily intended to support applications with the camera aimed near the horizon or oblique views but can also be used for nadir imagery.

Imagery requirements

Minimum requirements

This document presumes the existence of individual images with metadata describing the approximate camera location as well as orientation, hence the term *oriented* imagery. Additional metadata and more accurate orientation can be included for more advanced functionality. Geotagged imagery is defined as a subset of oriented imagery that has location but no orientation. It is supported but cannot provide any mensuration and can only provide limited navigation capabilities.

Camera support

Oriented Imagery is designed to support the following types of cameras:

- **Frame cameras.** This includes most drone and aerial imagery, typically with a field of view of about 70 degrees or smaller. Most digital cameras and video fall into this category.
- **Wide-angle or fish-eye cameras.** These typically have a greater than 70-degree field of view.

- **Omnidirectional cameras.** Panoramic, spherical, catadioptric, and fish-eye cameras are included in this category.
- **Synthetic omnidirectional cameras.** These have properties similar to an omnidirectional camera, but with a field of view typically created by processing the input from multiple images into a virtual image that typically has a predefined direction and distortion. These images are defined by methods including single frame, equirectangular images, and cube images.

Image structure

The images may be a single image, as in the case of a frame camera, or they may be made up of multiple images that need to be stitched together or images that have been processed to represent a single, stitched image (e.g., equirectangular images and cube images). Various stitching methods are supported.

Image format

A variety of formats are supported. These include simple browser-readable JPEG/PNG files, as well as formats such as MRF and COG that are optimized for cloud storage. Additionally, the imagery can be accessed from tile services or image services. Tile services provide a predefined tiling scheme where each tile can be accessed as a URL. Image services enable dynamic requests for a specified area of interest (AOI) and resolution and return only the pixels required.

Metadata

Metadata about the imagery is required. This not only includes the location of the imagery but also various attributes that define the orientation, view angles, and a wide range of attributes about each camera. A generic view of the data structure indicates that each image has a wide range of standardized parameters that define all properties, such as location, orientation, and image-specific metadata. Such metadata is provided as a point-based feature service that enables the image for an AOI to be efficiently queried and key attributes returned. In a typical scenario, the feature service is created from a collection of imagery and its attributes. Oriented imagery also provides support for image services sourced from a mosaic dataset of oblique imagery. This enables an image service to be directly referenced in an Oriented Imagery application, removing the requirement for a separate point-based feature service. Alternatively, an oriented imagery point-based feature service can define the image orientation with the imagery being read from an image service by making use of the ICS export capability.

Oriented imagery catalogs

Data structure

An oriented imagery catalog (OIC) is defined by a JSON data structure that defines key properties as well as a reference to the image-specific metadata. The key properties not only include generic parameters but also define defaults and variables. The use of the default and variables can significantly reduce the number of attributes required to define a collection of imagery with similar properties. If all the images in a collection of imagery have the same field of view, this can be defined in the OIC JSON file and need not be defined for each image. Similarly, if part of the URL to the imagery is always the same, this can be defined as a variable to reduce redundancy in the database.

Approximate versus accurate georeferencing

A key principle in the OI design is to enable both approximate and accurate georeferencing. The accuracy of the georeferencing is highly dependent on the calibration of the camera and knowing the camera parameters and auxiliary data, such as a digital terrain model (DTM) or depth image. To enable support for a wide range of accuracies while still enabling simple implementation, the georeferencing is split into two parts. One set of georeferencing is always required and provides georeferencing accuracy suitable for the determination of the appropriate image location and navigation. This provides approximations to the image2ground and ground2image transforms and is typically not sufficient for any mensuration. An optional second set of georeferencing can define accurate transformation that can be used to define accurate mensuration and associated accuracies.

Security

Security is handled at the OIC, feature service level and image levels. This means that the OIC JSON can contain all the information required to access the imagery and a link to the service that provides image-specific metadata. In a typical implementation, the OIC is defined as an item in ArcGIS Online (or your ArcGIS Enterprise portal) and users that have access to this can access the key properties including a reference to the feature (or image service) that defines the camera-specific data. This feature service may be a direct URL to a service or an item in ArcGIS Online (or your ArcGIS Enterprise portal) such that only users that have access to the service can access the imagery. This enables the inclusion of views or similar into the feature service so that users can access only specific subsets of the whole. It is typically assumed that if the OIC item can be accessed, then the imagery can also be accessed; i.e., the feature service will typically point directly to a URL that enables access to the imagery. It is possible to include additional authentication to control access to the imagery.

The client application therefore needs access to the OIC item, the feature service providing metadata, and the imagery. Access to the pixels can be directly from cloud storage or using a tile service API. One recommended way of providing access to the imagery is to put the data in an obfuscated cloud storage location such that access is not specifically authenticated but access to the URL is assumed to be equivalent to accessing the data source. Such a system enables massive volumes of imagery to be accessed while simplifying access control. Data access can also be handled by using attachments to features. If the images are defines as attachment to feature services then access control performed as part of access to the feature service.

Another method to handle security is by using a token server. The exposure points can be defined using variables that reference a token server. Refer to **Appendix A** below for more details for parameters that are supported and how to set up an OIC to use a token server to access your imagery.

Licensing

Oriented Imagery is controlled through the use of an OI API that is used in both web and desktop applications. This API limit capability depending on the user's subscription level in ArcGIS Online. Access to imagery and navigation is open. Access to simple mensuration requires the Viewer User Type. Access to a collection of features requires the Editor User Type. Access to any authoring capability requires the Creator User Type. Access to running more extensive analysis such as automated feature extraction requires ArcGIS Image for ArcGIS Online.

Modes

Oriented Imagery has many modes that are controlled by the OIC. This allows great flexibility while providing simplicity for the end users, but can create complexity for the authoring of OICs and the software development. The following information attempts to define the different modes.

The OIC can refer to a feature service or image service as the source of metadata that defines the location and orientation for all images. If defined by a feature service, the location of the images for navigation is determined by attributes of the service. The mensuration capabilities are determined based on availability of either approximate orientation or more accurate sensor-specific orientation..

If the source is an image service, then the navigation, image access, and mensuration are defined by the image service.

If the source is a feature service, then the imagery can be accessed multiple ways, including direct access to imagery in web-accessible storage, access using a tile server, or access via an image service. It is possible to *define* the geometry using a feature service, but *access* the pixels via image coordinate system (ICS) requests to an image service. In this case, the image service only needs to contain imagery, no georeferencing is required. Such an image service can be very quick to implement, provides on-the-fly clipping of the imagery, and can take advantage of the image service's access control.

As mentioned above, the georeferencing of the imagery for navigation purposes is defined in the feature by attributes. These provide only a limited level of accuracy. The attributes can also define a collection of sensor-specific parameters that can be used to provide more accurate mensuration.

Third dimension

Determining a relationship between image space and ground space requires a definition of three dimensions in both ground and image space as well as a transformation between them. A typical image does not have a third dimension, so one needs to be defined. This third dimension can be defined for each image in one of two ways:

DEM—Digital elevation model that is defined in map space. For any defined x,y location, the DEM should provide a height. Based on x,y,z + the ground2image transform, the appropriate pixel in image space can be identified. Having a DEM (and ground2image transform) for an image means that for any point on the map, the associated image location can be found. Going from image2ground is more complex and can only be achieved iteratively while needing to make assumptions on the form of the DEM. For many applications, no DEM is known, and ground needs to be approximated by the height of the camera above a flat terrain. If an oriented image is integrated with a 3D scene then the features of 3D scene can provide the 3rd dimension. Selecting a location in the scene (eg a 3D feature or point in a point cloud) returns a x,y,z coordinate which can be passed through the ground2image transform to return an image pixel. If an image location is selected the system can apply ray tracing to determine where the ray will intersect with the 3D scene.

Depth Image—For images that are collected at the same time as data is collected by a lidar system or other system that measures the suitable geometry between frames, a depth image can be defined. There are also deeplearning models that can estimate a depth image from a single image. If such an associated depth image exists, then the transform from image space to ground space is simplified to a single equation. A depth image is important for more accurate mensuration in complex terrains.

Oriented Imagery supports both methods of defining the third dimension.

Integration with artificial intelligence

In addition to providing navigation and mensuration capabilities for oriented imagery, OICs are also designed to be used as the source for applications requiring image-based feature identification. As a catalog, an OIC can be used to define what imagery should be submitted to artificial intelligence (AI) algorithms but, more importantly, to transform labels and features from image space to ground space for additional spatial processing.

Camera position and orientation

By default, the x,y position of the point feature defines the location of the camera in a suitable coordinate system. This may be a projected coordinate system, such as universal transverse Mercator (UTM) or web Mercator, or geographic coordinates. It is assumed that the y-axis is toward north.

The point feature location is primarily used to aid in the search for the appropriate images to display. As clarified later, the location of the camera may also be offset from this location or defined in a local datum to enable more accurate measurements.

If defined, the z-value of the point is assumed to be that of the camera location and may be defined by orthometric or ellipsoid height as part of the coordinate system. As will become apparent, the z-value is only used to show the location of the camera in 3D environments and is not necessarily used to define the location of objects in the image or the visible extent of the image.

The camera orientation is described in terms of **CamHeading**, **CamPitch**, and **CamRoll** ("cam" to clearly differentiate from orientation of the aircraft/vehicle body). These angles describe the camera orientation relative to a local projected coordinate system and refer to the point between the camera position and a point running through the center of the image.

Camera orientation is described as follows:

- The initial camera orientation is with the lens aimed at nadir (negative z-axis), with the top of the camera (columns of pixels) pointed north and rows of pixels in the sensor aligned with the x-axis of the coordinate system.
- The first rotation (**CamHeading**) is around the z-axis (optical axis of the lens), positive rotations **clockwise** (left-hand rule) from north.
- The second rotation (**CamPitch**) is around the x-axis of the camera (rows of pixels), positive **counterclockwise** (right-hand rule) starting at nadir.
- The final rotation (**CamRoll**) is a second rotation around the z-axis of the camera, positive **clockwise** (left-hand rule).

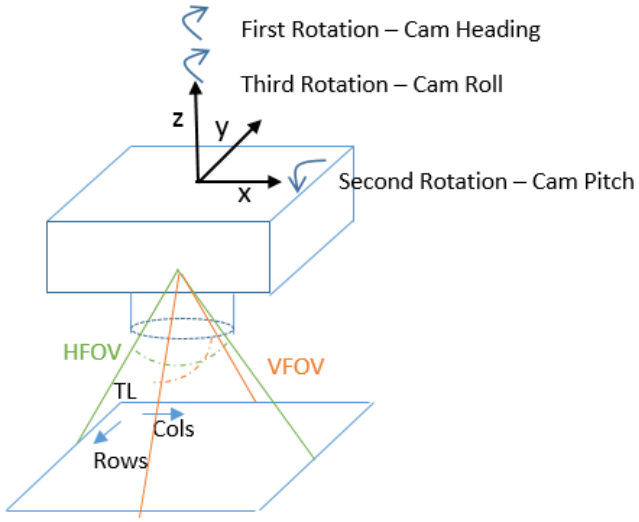


Figure 1

Such an orientation is simplest to comprehend. Assuming you are standing at the camera location looking north how much would need to rotate (heading) clockwise , then tilt the camera up (pitch) and the turn along the axis of the camera (roll) to point in the specified direction.

Example orientations

- The camera pointing down with the rows of pixels going from west to east would have the orientation 0,0,0.
- Rotating the camera 90 degrees so that the pixels go from north to south would be 90,0,0.
- Rotating the camera to the horizon would have the orientation 90,90,0.
- Rotating the camera counterclockwise by 20 degrees would result in an orientation of 90,90,20.

In most applications, the roll angle is 0. The roll angle is used to indicate that the camera body is rotated around the lens axis and is required to determine the correct pixel-to-image relationship.

In some cases, the image is rotated with respect to the camera. Consider taking a picture with most digital camera / mobile phones, even if you rotate the camera the resulting image is has up at the top of the image. This is handled by the ImgRot field that clarifies an additional rotation with respect to the camera. Horizontal field of view (HFOV) and vertical field of view (VFOV) should be that of the camera and should not change based on the roll angle.

Oriented imagery catalog schema

The structure of the oriented imagery catalog is summarized by the following:

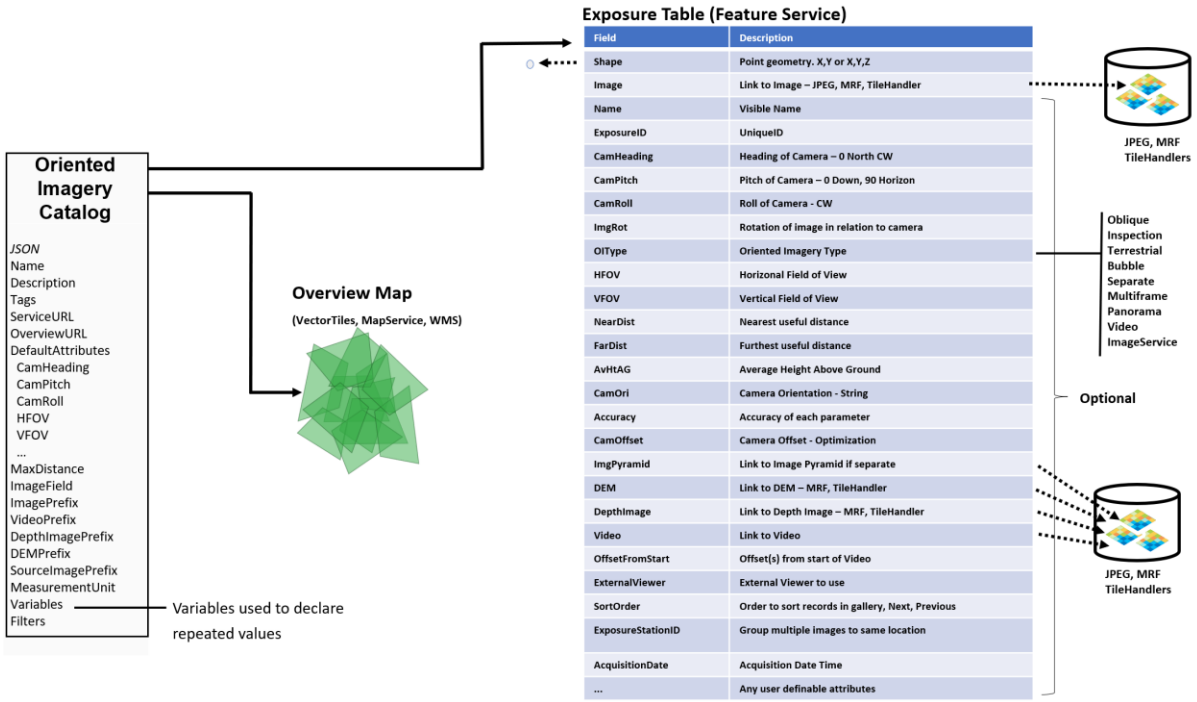


Figure 2

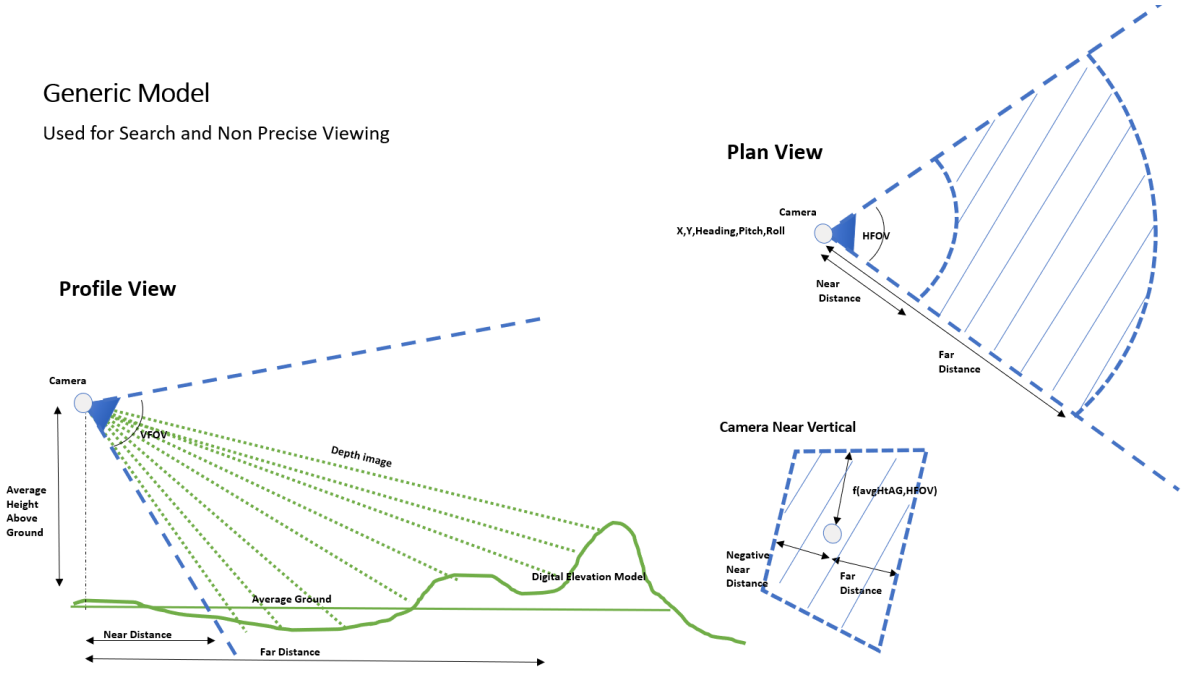


Figure 3

Oriented Imagery Exposure Table

This section defines field names, types, and default values in the feature class/service. Note that the primary use for this metadata is an efficient search capability to enable a user to quickly find and display images covering a site of interest and so includes a number of approximations. An optional CamOri field is provided to support precise measurements and can include accurate metadata (e.g., focal length, lens distortion) to enable more accurate measurements. This separation of approximate and precise metadata enables the support of an extensive range of imagery types.

Field name <default values in brackets>

- **Shape geometry X,Y—**Defined location of the camera. See also "CamOffset" attribute below. Optionally may also be x,y,z.
- **Name (optional)—**Name given to the raster and may be displayed for identification purposes. It may not be unique. It should not include the file extension.
- **ExposureID (optional)—**String maximum 50 characters—String that can be used as a unique identifier for the specific camera exposure. This is used to link any feature captured from the image back to the specific image. When features are captured using Oriented Imagery feature capture tools, the features are attributed with an Image URN that often contain this value. The Image URN value recorded is determined as follows: If the Exposure Table (that forms the OIC) has a GUID, then this should be used. Else, if the OIC contains an ExposureID field, then this should be used. Otherwise, the Image URN is set as follows:

  Name + "|" +
  Right (Name+"|"+ServiceName+"|"ObjectID, 50)
  where ServiceName is the service name (excluding \feature\...)
  Note: if Name is undefined, then this would be right("|"+ServiceName+"|"ObjectID, 50)
  On the creation of an OIC, some sources may already have unique IDs that are used to uniquely identify the image and so can be put in the ExposureID field.

- **Image—**Reference to the image. This can be one of the following:
  - URL to a web accessible image file (including extension) in cloud storage. Initial supported formats are: JPEG, TIFF, MRF. MRF provides a cloud-optimized, multiresolution, tiled format for faster access. JPEG may also be used but may not be optimum, especially for larger files. See notes below on ImgPyramid field that can be used to provide multiresolution access.
  - Local file path to a JPEG and TIFF image (Works only in ArcGIS Pro Add-In).
  - In some managed (database only) implementations, image may refer to a binary large object (BLOB) in which the image is stored. This may be applicable in offline mobile type applications.
  - May also refer to different services. Currently, four services are supported. The Image value needs to be prefixed with an identifier that indicates the type of file service and the schema used. The following notation should be used:

    ArcGIS Image Server
    A|ImageServiceURL|ObjectID|Col|Row—ArcGIS Image Server.
    A—Identifier.
    ImageServiceURL—URL to the image services (excluding Export).
    ObjectID—ObjectID of the image. The API will construct the URL to enable specified rows/columns to be returned.
    Col—Number of columns (image width).

Row—Number of rows (image height).

Tile Service
T|ZSTS|Col|Row|URL or T|ZSTS|URL.
T—Identifier.
ZSTS—Defines scale factor, number of levels, and tile size. For example, 42512 defines that there are 4 levels of zoom each with a factor of 2, and the tile size is 512. Only the following are supported: Number of Levels 1–9; zoom factors 2,3; tile sizes 256,512.
Col—Number of columns (image width) (In case of cube images, specify the width of one face in cube).
Row—Number of rows (image height).
URL—URL to the tile service using the following notation to define the location of a tile:
|z|—zoomLevel
|x|—col
|y|—row
|f| or |F|—For cubemap (360) images. F denotes the face of the cube. Valid values are f,r,b,l,u,d or F,R,B,L,U,D.

Example URL: Tile url—esri.com/Tiles/ImageID/|z|/|x|/|y|.jpg
The Oriented Imagery API will replace |z| with the valid zoom level value, |x| with the appropriate column value, and |y| with the row value.

Feature Image Attachments
FA
FA – Identifier
Note: No URL is required
Attachments are images that are included on feature layers. Attachments allow you to store images in the geodatabase and access the images in more ways.

Token Server URL
S|ImagePath
Note: See Appendix A for more information on implementing a token server to access images.

- **AcquisitionDate (optional)—**DATE field. An important attribute for many applications. Also used to support temporal imagery search or filtering. Allows optional recording of date only or may also include time of day, as required by the application. Note that all dates and times should be relative to UTC.
- **CamHeading** <default = -999>. Defined above, in degrees. CamHeading = 0 means the top of the camera is oriented toward the north, or CamHeading = 90 means the top of the camera is oriented toward the east. Value should be between 0 and 360. -999 or NULL indicates that orientation is not known, such as for georeferenced imagery.
- **CamPitch** <default = 90>. Defined above, in degrees. CamPitch = 90 means the camera is viewing toward the horizon. 0 indicates straight down.
    - The default value is based on the context of "inspection/street view" (non-nadir mapping) imagery.
    - Drone imagery may be at a range of oblique angles and, if nadir, would require a value of 0.

    - For inspection imagery, CamPitch > 90 may occur.
    - A negative value is allowed but is equivalent to using CamHeading + 180 and CamPitch + 180.
- **CamRoll** <0>. Defined above, measured in degrees. CamRoll will typically be near 0. CamRoll is used to account for +/- 90-degree rotations in the camera housing.
- **ImgRot—**Orientation of the image. The orientation of the camera relative to the scene, when the image was captured. Default is <0>. This rotation is added in addition to Roll. Value can be from 0 to 360. Negative values are allowed.
- **HFOV** <60>. Horizontal field of view, measured in degrees. This typically refers to the FOV for the row direction of the camera but may be the column direction if the CamRoll is +/- 90. The value can be estimated based on the focal length of the camera and CCD (image sensor) width.
    - HFOV ≈ 2*atan(sensor_width/(2*focallength))
- **VFOV** <40>. Vertical field of view, measured in degrees. Typically refers to the HOV for the column direction of the image but may not be if CamRoll is +/- 90. The value can be estimated based on the focal length of the camera and CCD height.
    - VFOV ≈ 2*atan(sensor_height/(2*focallength))
- **AvgHtAG** <1.8>. Average height (m) above ground of the camera. For aerial or drone imagery, this is equivalent to the flying height above ground. For indoor imagery, this is the height above the appropriate floor. For inspection imagery, it is the height above the base of the appropriate feature. The value is used to determine the visible extent of the image, so large values will result in a greater view extent. It is also used to help estimate the location of an object of known relative coordinates in the image. It can be used to enable identification of an appropriate image when using multiple images of tall objects.
- **FarDist** <20>. Furthest usable distance from the camera position of the imagery in meters.
    - This value is used for searching on a 2D map; e.g., click on the map, "Do I have an image that shows this location?" Beyond this distance, the answer may be technically yes, but it's beyond the distance for the image to be usable for detailed inspection or interpretation (a judgment based on the application).
    - For close-range applications, the value is typically set manually versus calculated; e.g., for close-range inspection, this may be ~5 m; for street view imagery, perhaps ~20 m; for structural inspection/damage assessment from a drone, ~50 m.
    - If nadir, then set FarDist = AvgHtAG.
    - FarDist should always be >0.
- **NearDist** <0>. Nearest usable distance of the imagery in meters. Near distance can never be less than 0.
    - If nadir, then set NearDist = 0.
- **OIType (optional**)**—**Oriented Imagery type, string to define type of imagery
    - **O—**Oblique—Frame image from aerial or drone sensor
    - **I—**Inspection—Frame image with feature of interest near camera
    - **T—**Terrestrial (Default)—Typical frame imagery taken at street level
    - **B—**Bubble—A single stitched panorama image with 360-degree view (equirectangular image—photosphere)

Figure 3

- **S—**Six separate images stitched together to form a cube. Order of images should be [Front, Right, Back, Left, Down, Up], and front face of the cube should have CamHeading = 0; Cubemap image—a cube with six faces.
  Example

Figure 4

- **D—**Multiple frame images taken from the same location, pointing in different directions but considered a group. Order of images should be [Front, Right, Back, Left, Down, Up], and front face of the cube should have CamHeading = 0; Cubemap image—a cube with six faces.
  Example—Image URL of front face (provided in Image field):
  http://s3.amazonaws.com/sample_F.png
  It should have _F at the end. API will append a character at the end of the string to get all the faces of the cube.
  For Front, it will create the URL http://s3.amazonaws.com/sample_F.png.
  For Right, it will create the URL http://s3.amazonaws.com/sample_R.png.
  For Back, it will create the URL http://s3.amazonaws.com/sample_B.png.
  For Left, it will create the URL http://s3.amazonaws.com/sample_L.png.
  For Down, it will create the URL http://s3.amazonaws.com/sample_D.png.
  For Up, it will create the URL http://s3.amazonaws.com/sample_U.png.
  These URLs should exist in the same location.
  (Note that these names may change.)
- **P—**Panorama—HFOV >4x VFOV. Equirectangular—photosphere. HFOV > 150 && HFOV < 360

Figure 5

  - **V—**Video. Image may or may not exist. Video field should define location of video.

  This setting affects some functionality in the User Interface components.
- **SortOrder—**Optional LongInteger. A number used to define the order of the imagery and to enable gallery type applications, where users make a selection and go from one image to the next, typically along a street or down a power line or pipeline. If undefined, then AcquisitionDate is used so long as it exists and includes time; otherwise, ObjectID is used.
- **CamOffset (optional)—**String of two or three comma-delimited values (dx,dy,dz) that define the actual camera location relative to the feature point. The use of this is primarily for oblique imagery with longer focal lengths. It enables smaller search distances to be used when identifying a suitable image for a location. When determining the extent and positioning of the camera, the application must add these values to the feature location to determine the camera location. Note that the units are defined in that of the feature service; i.e., if the service is measured in feet, then this unit is in feet.
- **Accuracy (optional)—**Comma-delimited string of values with the following order: StdDevXY(m), StdDevZ(m), StdDevHeading(deg), StdDevPitch(deg), StdDevRoll(deg), StdDevDistNear(m), StdDevDistFar(m), StdDevElevation(m). Default or 0 defines unknown. These are used to indicate the accuracy of the orientation measurements and to control how the location cursors are displayed. If undefined, the position cursors indicate very approximate positions or may not enable measurement capabilties. StdDevXY,Z are the standard deviation of the camera location accuracy. Eg a GPS may have a position of +/- 10m RMS in XY and +/- 20m in Height in which case the value would be 10,20. StdDevHead,Pitch,Roll are the standard deviation accuracy of angles. StdDevDistNear,Far are the standard deviation accuracy of the distance measurements for measurements at the near distance and far distance when using a depth image. It is assumed that measurement accuracy changes linearly with distance.
- **ExposureStationID (optional)—**Long integer that can be used to associate multiple images taken at the same location. This can be used to modes where panning should transition from image to image without moving to a different location. It is also used to assist in some editing functions such as if the location of the camera is moved that is part a group with the same ExposureStationID then all the cameras are moved.
- **ImgPyramids (optional)—**Optimization field providing information on the approximate size of the image and the possible availability of reduced-resolution JPEG versions that can be used to enable faster access to thumbnails of the full image. The field is a comma-delimited string of integers defining the approximate number of columns of the image width and availability of a reduced-resolution image. The image size is defined as the round (cols /100); i.e., an image with 5,550 columns would be defined as 55.5. Subsequent numbers define the availability, approximate size, and name of reduced-resolution versions. The URL to the reduced-resolution versions are assumed to be the same as the full-resolution version but with the file extension replaced by _XXXX.jpg. For example, if the Image was http://a.com/img/D023.jpg

[s3.amazonaws.com] and had a size of 5,550 columns, then a string of the form 55.5,20,5 would indicate the following images also exist:
http://a.com/img/D023_20.jpg [s3.amazonaws.com] has 2,000 columns,
http://a.com/img/D023_5.jpg [s3.amazonaws.com] has 500 columns.
The application can now access the appropriate image for display depending on the display window size and zoom ratio.
Note that although MRF | TIF files will typically have internal pyramids, such small JPEG files may still be used to increase performance, as small JPEG files can be accessed as single get requests.

- **ImgPyramidsUrl (optional)—**Optimization field providing reference (URL) for each reduced-resolution JPEG versions that can be used to enable faster access to thumbnails of the full image. The field should be used in conjunction with ImgPyramids field. The field is a comma-delimited string of URLs defining the location of each reduced-resolution image defined in ImgPyramids field. For example, if the ImgPyramids was 55,20,5, then a string of "http://a.com/img/ 20.jpg [s3.amazonaws.com],http://abcd.com/img/5.jpg [abcd.com]" would indicate that
Image with 2000 cols can be accessed using http://a.com/img/ 20.jpg [s3.amazonaws.com] URL, and
Image with 500 cols can be accessed using http://abcd.com/img/5.jpg [abcd.com] URL.
Image with 5500 cols would be accessed using the value provided in Image Field.

- **DEM (optional)—**Defines the Digital Elevation Model to be used to compute the ground to image transform for cases where no height information is available. Note that if a depth image exists it will be used for all image to ground transforms, but if one does not exist then the DEM will be used.

  ArcGIS ImageServer for Height
  I|PS|ImageServiceURL—ArcGIS Image Server|RenderingRule
  I—Identifier
  PS—Pixel size in meters
  ImageServiceURL—URL to the image services (excluding Export)
  Based on the coverage, the system should extract a terrain model of defined pixel size covering the possible image extent and use that to define elevation.
  RenderingRule (optional)—Rendering Rule defined on the service

  ArcGIS Tile ImageServer for Height
  E|ImageServiceUrl—ArcGIS Cached Image Services|LOD
  E—Identifier
  ImageServiceUrl—URL to the cached image service
  LOD (optional)—Level of Detail (max LOD Level). Tile will be retrieved from this level or below.

  MRF|LERC for Height
  URL—URL to MRF|LERC file
  Source can be image service, tile cache image service, MRF, or LERC.

- **DepthImg (optional)—**Used to define the depth dimension. Depending on the application, different methods are appropriate. Defining a depth image has a significant effect on the

measurement taken from the image. If Undefined or 0, then distance of the object from the camera is computed purely from AvgHtAG and camera angles, which will assume the ground is flat (or will use the DEM described above if defined).
The default notation is to define a URL to a web accessible image that provided depth information for the pixels. The extent of the image is assumed to be the same extent as the primary image, although it may have a different number of rows and columns. This URL may differ from image to image, or many images may reference the same depth image.

The following notations are used to define different ways of representing plane depth:

MRF|LERC|PNG for Depth
URL—URL to MRF|LERC|PNG file
The use of MRF is recommended. If using PNG then a specific formula is used to combine the three channels into 1.

Simple Plane
P|A,H,V
P- Identifier
Define distance as P|A,H,V where A is distance to center of image and H and V are the vertical and horizontal deviations (in deg) from a plane that is parallel to the camera. Currently, we don't support both H and V parameters together. In most cases V=0. Hence, we only use one of the angles with preference to the Vertical angle.
Distance is computed as
If V != 0
$D = A * ((\cos(V-V_{ang}) + \sin(V_{ang}) * \sin(-V))/(\cos(V- V_{ang}) * \cos(V_{ang}) * \cos(H_{ang})))$
Else if H != 0
$D = A * ((\cos(H-H_{ang}) + \sin(H_{ang}) * \sin(-H))/(\cos(H- H_{ang}) * \cos(H_{ang}) * \cos(V_{ang})))$
Else
$D = A / (\cos(H_{ang}) * \cos(V_{ang}))$
Where H_{ang} = HFOV*Abs(Col-(NCols/2)), V_{ang} = VFOV*Abs(Row-(NRows/2))

TileService
T|ZSTS|URL
T—Identifier
ZSTS—Defines scale factor, number of levels, and tile size
Example: 42512 defines that there are 4 levels of zoom, each with a factor of 2, and the tile size is 512. Only the following are supported: Number of Levels 1–9; zoom factors 2,3; tile sizes 256,512.
URL is URL to tile service using the following notation to define the location of tile:
|z|—zoomLevel
|x|—col
|y|—row
|f| or |F|—For bubble images. F denotes the face of the cube. Valid values are f,r,b,l,u,d or F,R,B,L,U,D.

Example: Tile url—https://esri.com/PanoramaTiles/ImageID/|z|/|x|/|y|.lrc

The Oriented Imagery API will replace |z|with valid zoom level value, |x| with appropriate column value, and |y| with row value.

It is often possible to define a depth image as a single value (e.g., some inspection imagery) or a plane (e.g., oblique imagery of a flat terrain), but this is not yet supported.

- **CamOri (optional)—**Detailed camera orientation, stored as a comma-delimited string. This field provides support for more accurate measurements, depending on the application.
  The first number indicates the type. Over time, many different types may exist. Some applications may only support some of the types.

  Type 1—Heading, Pitch, Roll
  Value ordering:
  1|WKID_H|WKID_V|X|Y|Z|H|P|R
  or
  1|WKID_H|WKID_V|X|Y|Z|H|P|R|A0|A1|A2|B0|B1|B2|FL|PPX|PPY|K1|K2|K3|P1|P2
    - WKID_H—WKID for horizontal coordinate system
    - WKID_V—WKID for vertical coordinate system. Can be undefined but must be same unit as the WKID_H. 115700 is Ellipsoidal and 105700 is Orthometric
    - X,Y,Z—Camera center coordinate (perspective point)
    - H,P,R—Heading, pitch, roll in degrees—Define as in key attributes
    - A0 A1 A2 B0 B1 B2 Affine transformation parameters to camera center in the ground to image direction (i.e., A0 and B0 are offsets in cols and rows, and A1,B2|A2,B1 are 1/pixel size in microns.)
    - FL—FocalLength in microns
    - PPX,PPY—PrincipleX,PrincipleX—Principal point offset in X and Y from camera center in microns
    - K1,K2,K3—Conrady (radial) distortion coefficients
    - P1,P2—Tangential distortion coefficients

  Note – FL, PPAX,PPAY and Affine transform (A1/A2, B1/B2) must have same unit. Unit can be in microns, pixels, or mm.

  Type 2—Omega, Phi, Kappa
  Value ordering:
  2|WKID_H|WKID_V|X|Y|Z|O|P|K
  or
  2|WKID_H|WKID_V|X|Y|Z|O|P|K|A0|A1|A2|B0|B1|B2|FL|PPX|PPY|K1|K2|K3|P1|P2
    - WKID_H—WKID for horizontal coordinate system of the point (EPSG)
    - WKID_V—WKID for vertical coordinate system of the point
    - X,Y,Z—Camera center coordinate (perspective point)
    - O,P,K—Omega, phi, kappa in degrees
    - A0 A1 A2 B0 B1 B2—Affine transformation parameters to camera center (i.e., A0 and B0 are offsets in cols and rows, and A1,B2|A2,B1 are 1/pixel size in microns.)
    - FL—FocalLength in microns
    - PPAX,PPAY—Principal point offset in X and Y from camera center in microns

- K1,K2,K3—Conrady distortion coefficients
- P1,P2—Tangential distortion coefficients

Note – FL, PPAX,PPAY and Affine transform (A1/A2, B1/B2) must have same unit. Unit can be in microns, pixels, or mm.

Type 3—Yaw, Pitch, Roll

Value ordering: 3|WKID_H|WKID_V|X|Y|Z|Y|P|R

- WKID_H—WKID for horizontal coordinate system of the point
- WKID_V—WKID for vertical coordinate system of the point
- X,Y,Z—Camera center coordinate (perspective point)
- Y,P,R—Yaw, Pitch, Roll in degrees

Note- Type 3 is currently only valid for 360 images.

Type 4—Local tangent plane (ENU) coordinates

Value ordering:

4|LTPCentreLat| LTPCentreLon|EllipsoidRadius|EllipsoidEE|1|WKID_V|LTPX|LTPY|LTPZ|H|P|R

or

4|LTPCentreLat|LTPCentreLon|EllipsoidRadius|EllipsoidEE|1|WKID_V|LTPX|LTPY|LTPZ|H|P|R|A0|A1|A2|B0|B1|B2|FL|PPX|PPY|K1|K2|K3|P1|P2

or

4|LTPCentreLat| LTPCentreLon|EllipsoidRadius|EllipsoidEE|2|WKID_V|LTPX|LTPY|LTPZ|O|P|K

or

4|LTPCentreLat|LTPCentreLon|EllipsoidRadius|EllipsoidEE|2|WKID_V|LTPX|LTPY|LTPZ|O|P|K|A0|A1|A2|B0|B1|B2|FL|PPX|PPY|K1|K2|K3|P1|P2

or

4|LTPCentreLat| LTPCentreLon|EllipsoidRadius|EllipsoidEE|3|WKID_V|LTPX|LTPY|LTPZ|Y|P|R

- WKID_V—WKID for vertical coordinate system of the point
- LTPX,LTPY,LTPZ—Camera center coordinate (perspective point) in LTP space ENU
- O,P,K—Omega, phi, kappa in degrees
- H,P,R—Heading, pitch, roll in degrees—Define as in key attributes
- Y,P,R—Yaw, Pitch, Roll in degrees
- A0 A1 A2 B0 B1 B2—Affine transformation parameters to camera center (i.e., A0 and B0 are offsets in cols and rows, and A1,B2|A2,B1 are 1/pixel size in microns.)
- FL—FocalLength in microns
- PPAX,PPAY—Principal point offset in X and Y from camera center in microns
- K1,K2,K3—Conrady distortion coefficients
- P1,P2—Tangential distortion coefficients
- LTPCentreLat – the latitude in degrees of the LTP coordinate system origin
- LTPCentreLon – the longitude in degrees of the LTP coordinate system origin
- EllipsoidRadius – Earth's equatorial radius in meters
- EllipsoidEE – Eccentricity squared

Read more about Local tangent plane here.

Orientation angles supported in Oriented Imagery

CamHeading, CamPitch, CamRoll

The initial camera orientation is with the lens aimed at nadir (negative z-axis), with the top of the camera pointed north and rows of pixels in the sensor aligned with the x-axis of the coordinate system.

- The first rotation (**CamHeading**) is around the z-axis (optical axis of the lens), positive rotations **clockwise** (left-hand rule) from north.
- The second rotation (**CamPitch**) is around the x-axis of the camera (rows of pixels), positive **counterclockwise** (right-hand rule) starting at nadir.
- The final rotation (**CamRoll**) is a second rotation around the z-axis of the camera, positive **clockwise** (left-hand rule).

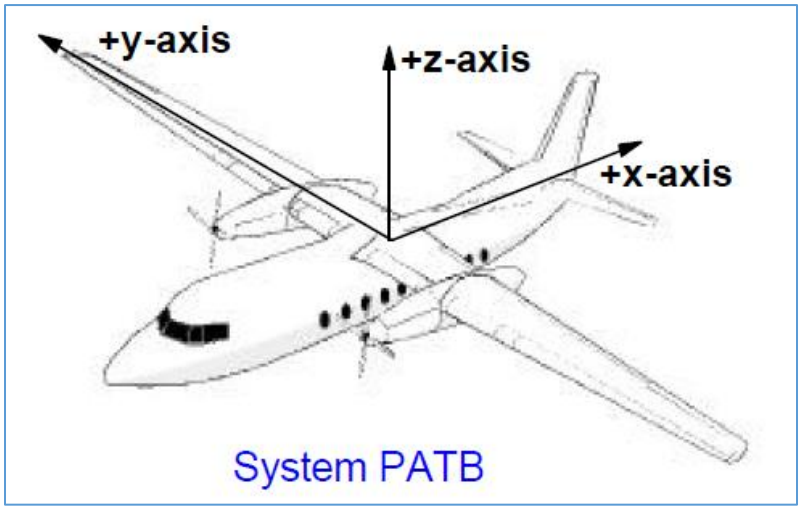
For rotations performed in the sequence of CamHeading (ψ) first, then CamPitch (θ), then CamRoll (ϕ), the resulting rotation matrix **R** is

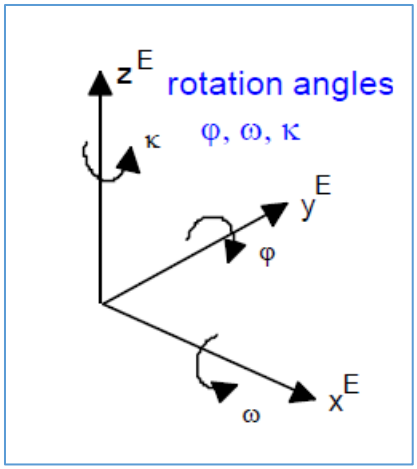
$$\mathbf{R = R_z(\psi)R_x(\theta)R_z(\phi)}$$

$$\mathbf{R} = \begin{vmatrix} \cos\psi * \cos\phi - \cos\theta * \sin\psi * \sin\phi & \cos\psi * \sin\phi + \sin\psi * \cos\theta * \cos\phi & -\sin\psi * \sin\theta \\ -\sin\psi * \cos\phi - \cos\psi * \cos\theta * \sin\phi & \cos\psi * \cos\theta * \cos\phi - \sin\psi * \sin\phi & -\sin\theta * \cos\psi \\ -\sin\theta * \sin\phi & \sin\theta * \cos\phi & \cos\theta \end{vmatrix}$$

Omega, Phi, Kappa

Omega, Phi, Kappa angles define the rotation between a projected coordinate system and the image coordinate system. It is mostly used in photogrammetry software.





All rotations are measured positive counterclockwise. For rotations performed in the sequence of Omega first, then Phi, then Kappa, the resulting rotation matrix **R** is

$$\mathbf{R = R_X(\omega)R_y(\varphi)R_z(\kappa)}$$

$$\mathbf{R} = \begin{bmatrix} \cos\varphi \cdot \cos\varkappa & -\cos\varphi \cdot \sin\varkappa & \sin\varphi \\ \cos\omega \cdot \sin\varkappa + \sin\omega \cdot \sin\varphi \cdot \cos\varkappa & \cos\omega \cdot \cos\varkappa - \sin\omega \cdot \sin\varphi \cdot \sin\varkappa & -\sin\omega \cdot \cos\varphi \\ \sin\omega \cdot \sin\varkappa - \cos\omega \cdot \sin\varphi \cdot \cos\varkappa & \sin\omega \cdot \cos\varkappa + \cos\omega \cdot \sin\varphi \cdot \sin\varkappa & \cos\omega \cdot \cos\varphi \end{bmatrix}$$

Yaw, Pitch, Roll

Yaw (ψ) is measured positive clockwise while Roll (θ) and Pitch (ϕ) are measured positive counterclockwise. For rotations performed in the sequence of Yaw first, then Roll, then Pitch, the resulting rotation matrix **R** is

$$\mathbf{R = R_z(\psi)R_y(\theta)R_x(\phi)}$$

$$\mathbf{R} = \begin{vmatrix} \cos\psi * \cos\theta & \sin\psi * \cos\phi + \cos\psi * \sin\theta * \sin\phi & \cos\psi * \sin\theta * \cos\phi - \sin\psi * \sin\phi \\ -\sin\psi * \cos\theta & \cos\psi * \cos\phi - \sin\psi * \sin\theta * \sin\phi & -\sin\phi * \cos\psi - \sin\psi * \sin\theta * \cos\phi \\ -\sin\theta & \cos\theta * \sin\phi & \cos\theta * \cos\phi \end{vmatrix}$$

Rotation matrix R to Orientation angles

The above section clearly shows how to compute a rotation matrix R from the orientation angles defined in Oriented imagery. We can also compute same orientation angles from any given rotation matrix R.

Given **R**

$$\mathbf{R} = \begin{vmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{vmatrix}$$

CamHeading = ATAN2(- R_{13}, - R_{23}) * 180 / PI
CamPitch = ACOS(R_{33}) * 180 / PI
CamRoll = ATAN2(-R_{31}, R_{32}) * 180 / PI

Omega = ATAN2(- R_{23}, R_{33}) * 180 / PI
Phi = ASIN(R_{13}) * 180 / PI
Kappa = ATAN2(- R_{12}, R_{11}) * 180 / PI

Yaw = ATAN2(- R_{21} , R_{11}) * 180 / PI
Pitch = ATAN2(R_{33} , R_{32}) * 180 / PI
Roll = ASIN(- R_{31}) * 180 / PI

OIC properties

Each OIC is defined by a properties JSON that is referenced as a file or an item in ArcGIS Online. In the OIC management tools in ArcGIS Pro, this file also defines the key properties.

The file should have the extension .oic

Following is the data structure:

```
{
	"type": "OIC",
	"version": "1.0",
	"properties": {
		"Name": "",
		"Description": "",
		"Tags": "",
		"ServiceURL": "",
		"OverviewURL": "",
		"DefaultAttributes": {
			"CamHeading": "",
			"CamPitch": "",
			"CamRoll": "",
			"HFOV": "",
			"VFOV": "",
			"AvgHtAG": "",
			"FarDist": "",
			"NearDist": "",
			"OIType": "",
			"SortOrder": "",
```

```
                "CamOffset": "",
                "Accuracy": "",
                "ImgRot": "",
                "CamOri": "",
                "ImgPyramids": "",
                "DepthImg": "",
                “DEM”: “”
            },
            "DefaultImage": "",
            "DefaultSearchLocation": {},
            "About": "",
            "ImageField": "",
            "ImagePrefix": "",
            "ImageSuffix": "",
            "DepthImagePrefix": "",
            "DepthImageSuffix": "",
            "DEMPrefix": "",
            "DEMSuffix": "",
            "SourceImagePrefix": "",
            "CoveragePercent": 0,
            "MaxDistance": "",
            "MeasurementUnit": "meter",
            "TimeUnit": "days",
            "Credentials": {
                "Username": "",
                "Password": ""
            },
            "Variables": {},
            "Filters": {},
            "Copyright": {
                "text": "",
                "url": ""
            }
        }
}
```


Notes

Name—Name of the Oriented Imagery catalog.

Description—Short description of the Oriented imagery catalog.

Tags—Tags used to aid identification

ServiceURL—URL to the feature service that defines the points and attributes.

OverviewURL—URL to the vector tile cache, map service, Web Map Service (WMS), or Web Map Tile Service (WMTS) or service item used to provide an overview of the image coverage. This is used only as a display of coverage and is not queried.

DefaultAttributes—Define defaults for each field if undefined or NULL in the feature class.

About—Help document. The format of the file should be .html.

DefaultImage- OBJECTID for any record in the exposure point table. If defined, viewer loads the specific image by default. For instance, if DefaultImage value is 1000, then the viewer loads the image by default with OBJECTID = 1000 from the exposure point table.
DefaultSearchLocation – JSON Point format. If defined, viewer searches for images which contains the search location and displays the best image by default. No need to select a point on the map / scene to view an image in the viewer.
JSON Point format ->

```
{
 "x": <x>,
 "y": <y>,
 "z": <z>,
 "spatialReference": {
  <spatialReference>
 }
}
```

For more information click here.
For example, if DefaultSearchLocation : {

```
"x": -118.15,
"y": 33.80,
"z": 10.0,
"spatialReference": {
        "wkid": 4326
}
}
```

Viewer will search for images that contains the search location provided above and load the best image by default.
ImageField—If defined, then the name of the field to be used as the Image Field
ImagePrefix—Prefix to be added to start of value in Image attribute. Typically used to define the root URL for the image.
ImageSuffix—Suffix to be added to end of value in Image attribute. Typically used to define the parameters at the end of the URL for the image.
DepthImagePrefix—Prefix to be added to start of value in DepthImg attribute. Typically used to define the root URL for the depth image.
DepthImageSuffix—Suffix to be added to end of value in DepthImg attribute. Typically used to define the parameters at the end of the URL for the depth image.
DEMPrefix—Prefix to be added to start of DEM attribute. Typically used to define the root URL for the DEM.
DEMPrefix—Suffix to be added to end of DEM attribute. Typically used to define the parameters at the end of the URL for the DEM.
SourceImagePrefix—Prefix to be added to start of source image name. Typically used to define the local network or path where the images are stored. Does not exist as part of the service.
CoveragePercent- Valid values are from -50 to 50. Default is 0. Based on the value provided, API will shrink or expand the ground footprint of each image which is computed to search for images that contains the location (red cross). -30 value will shrink the size of the footprint by 30% from the edges

and 20 value will increase the size of the footprint by 20%. This aids in removing the points that are right at the very edge of the image.
MaxDistance—Maximum search distance to be used while querying the service specified in ServiceURL.
MeasurementUnit—Valid values are meter or feet. Mensuration results will be shown in the unit defined here. Default is m.
TimeUnit—Valid values are days, weeks, months, or years. Time selector tool in the viewer will filter images based on the TimeUnit value defined here.
Credentials—Credentials required to access imagery served using TileService. Basic authentication method is used to access such imagery.
Copyright—Define copyright/attribute text to display on image

Filters—There is often a necessity to subset an OIC. A typical example is the street view imagery taken over multiple years where the user wants to see imagery only for a specific year. Another example may be imagery for a building at different floors, where users want to see only imagery for a specific floor. This can be handled by defining a set of filters in the OIC JSON as follows:

```
{
        "filters": {
                "All": "1=1",
                “AZ”: “State = 'AZ'”,
                "2016": "Where Year = 2016",        //* Replace these with appropriate statements *//
                "2014": "Where Year = 2014",
                "Old": "Where Year<2014"
        }
}
```

The OIC API will list the defined filters in their respective order and labels.

Variables—The number of records in the feature service can become very large, and there are many cases where parameters may need to be changed depending on a user. An example is an access key that may be embedded as part of a URL. This key may need to change over time or be specific to a user. To enable this, the concept of variables is used.
Any string attribute value may have one or more variables defined using the notation $_Variable_$; e.g., URL = $_Path_$/MyDirectory/A.JPG?/$_tag_$
As each attribute is read, any text between $_ and _$ will be replaced by a variable in the OIC. If no corresponding variable is defined, it is excluded.
Note: Variables should not be used in attributes that are searched, e.g., AcquisitionDate.

Variables are defined as JSON as follows:

```
{
        "variables": {
                "cyclo": {
                        "wrapperUrl": " ",
                        "config": {
                                "username": "aad",
                                "password": "asdasd",
                                "apiKey": "asdasdasdasdasd"
```

```
                    }
                },
                "path": "C:/temp"
        }
}
```

Defaults are values to be used in place of attribute fields. Each image is initiated with these default values, and then the defaults are replaced by any value in the attribute table.

This provides a much more compact method of defining OIC, as only the values that vary from camera to camera need to be defined in the attribute table.

Note that values such as AcquisitionDate do not have defaults, as they are used as query fields.

These properties need to be refined. Names will be refined based on ArcGIS Online naming conventions.

Appendix A: Token server implementation to access images on a secured server.

Additional security can be provided while using an oriented imagery catalog by storing the images on a secured server. The parameters required to access the secured imagery data can be defined under tokenserver (within the variables parameter) in the OIC JSON.

Workflow

- Create an oriented imagery catalog using the Oriented Imagery Management Tools.
- In the Image field, be sure to prefix the path to the image in the attribute table with "S|".
- Create a gateway (response) URL pointing to the secured server where the images are stored.
- Edit the OIC JSON and define the ***tokenserver*** parameters within the variables node (see below for example parameters).
  Note: Any parameter required to access the images can be defined within the tokenserver parameters and appended to the response URL. This includes the image URLs taken from the image fields in the attribute table, tokens to access the image, etc. If you plan to use values from the OIC feature service attribute table, you can define the relevant field name using the key word format $$feature.fieldName$$.

The following variables are defined for the process –

tokenserver : Define the parameters required to access imagery data stored in the secured server.

> **url** : The response URL (gateway URL) pointing to the secured server, to which the token server parameters are appended.
>
> **parameters** : Parameters define the input parameters to the URL that is used to access the tokenserver. This contains the list of all parameters that will be appended onto the response URL. The following sample parameters are defined in the OIC json snippet above:

id : Is a unique identifier that maybe used by the tokenserver to identify a user or dataset etc. In the example above the ID that is set to "**oisecuritytest"**

imageURL : This variable defines the column within the OIC feature service attribute table, which contains the image path. (It will always be the Image field.)

- Note: The image path need only be a partial path to the image. The token server URL path is used as a prefix to make a complete path that is then fed to the tokenserver for validation. $$feature.image$$ is a key word denoting the field name to be used.

  Sample image attribute value -
  S|orientedimagery/EsriCampus180701/109/IMG_7716_3629922.JPG

  where S| is a mandatory identifier to indicate that a token server is being used to access the image, followed by the folder and image name signifying its location.

  groupname : is a custom value used for this particular (sample) implementation of tokenserver. This can be replaced by any other name as required by the token server in use.

  ref : is a custom value for this particular (sample) implementation of tokenserver. This can be replaced by any other name as required by token server.

  rt : This defines the return type for the request that is sent to the token server. The following URL return types are currently supported:

  - "**presigned-url**": This return type is used when the complete image URL is available in the Image field (in the OIC feature service attribute table). This return type can be used to access the image directly as the API doesn't resolve or append to it.

  - "**image**": This return type is used when part of the image URL is available in the image field (in the attribute table) and the rest of the URL is split into a prefix/suffix within the OIC JSON. While using this return type, the API will add both prefix and suffix to the image URL and then return the complete URL.

headers : HTTP headers are used to pass additional information between client and server through request and response headers. Any additional header information can be added here.

method : HTTP method used to request data from a specified resource. Tokenserver implementation always uses "GET" method.

A snippet of the tokenserver implementation is shown below:

a. Return type (rt): **image**


```
"About": "",
"ImageField": "",
"ImagePrefix": "https://s3.us-east-1.amazonaws.com/",
"ImageSuffix": "",
"VideoPrefix": "",
"VideoSuffix": "",
"DepthImagePrefix": "",
"DepthImageSuffix": "",
"SourceImagePrefix": "",
"SourceImageSuffix": "",
"MaxDistance": "50",
"DEMPrefix": "",
"Credentials": {
    "Username": "",
    "Password": ""
},
"Variables": {
    "tokenserver": {
        "url": "https://mt0v0298r7.execute-api.us-east-1.amazonaws.com/dev/presigned-url",
        "parameters": {
            "id": "oisecuritytest",
            "imageURL": "$$feature.image$$",
            "groupname": "$$feature.Name$$",
            "ref": "test",
            "rt": "image"
        },
        "headers": {},
        "method": "get"
    },
    "abc": {}
},
```


b. Return type (rt): **presigned_url**


```
"About": "",
"ImageField": "",
"ImagePrefix": "",
"ImageSuffix": "",
"VideoPrefix": "",
"VideoSuffix": "",
"DepthImagePrefix": "",
"DepthImageSuffix": "",
"SourceImagePrefix": "",
"SourceImageSuffix": "",
"MaxDistance": "50",
"DEMPrefix": "",
"Credentials": {
    "Username": "",
    "Password": ""
},
"Variables": {
    "tokenserver": {
        "url": "https://mt0v0298r7.execute-api.us-east-1.amazonaws.com/dev/presigned-url",
        "parameters": {
            "id": "oisecuritytest",
            "imageURL": "$$feature.image$$",
            "groupname": "$$feature.Name$$",
            "ref": "test",
            "rt": "presigned_url"
        },
        "headers": {},
        "method": "get"
    },
    "abc": {}
}
```

Example call using the above snippet should give the following url:

url?id=oisecuritytest&imageURL=$$feature.image$$&groupname=$$feature.Name$$&ref=test&rt=image

https://sampletest.amazonaws.com/dev/presigned-url?id=oisecuritytest&imageURL=S|orientedimagery/EsriCampus180701/109/IMG_7716_3629922.JPG&groupname=109/IMG_7716_3629922$$&ref=test&rt=image

Appendix B: Embedding exposure points and coverage into the OIC JSON

An oriented Imagery catalog is defined as a JSON that references a point-based feature service that defines the camera location and orientation as well as other metadata. Traditionally, three items or files are created for every catalog—the OIC JSON, a feature service, and an optional coverage map (usually a vector tile service).

Using the feature service to manage images allows for scaling to many millions of images in a single catalog. However, it is not an optimum solution if the catalog has less than 100 images. Management of three separate items for a small set of images is cumbersome. Also, there is always a dependency on a feature service, which in some cases is not ideal.

The above issues are overcome by using this implementation of the OIC JSON, which supports GeoJSON features. By embedding both the GeoJSON exposure point features and GeoJSON coverage feature(s) in a single file, better management of small catalogs is enabled. This will also make small OICs more portable and easier to edit.

GeoJSON schema

The schema definition for embedding GeoJSON features and coverage in the OIC JSON is as below. It relied on items called GeoJSONFeatures and GeoJSONCoverage in the properties node. The names of these items should then be added to the **ServiceURL** and **OverviewURL** properties.

```
{
	"type": "OIC",
	"version": "1.0",
	"properties": {
		"Name": "Building_E_QuickCapture",
		"Description": "Oriented Imagery Catalog for QuickCapture project",
		"Tags": "OIC, OrientedImagery, QuickCapture, ArcGIS, Imagery",
		"ServiceURL": "GeoJSONFeatures",
		"OverviewURL": "GeoJSONCoverage",
		"DefaultAttributes": {
			"camheading": "-999",
```

```
                        "campitch": "90",
                        "camroll": "0",
                        "hfov": "60",
                        "vfov": "40",
                        "avghtag": "1.7",
                        "fardist": "50",
                        "neardist": "1",
                        "oitype": "T",
                        "sortorder": "",
                        "camoffset": "",
                        "accuracy": "",
                        "imgpyramids": "",
                        "depthimg": "",
                        "externalviewer": "",
                        "imgrot": ""
                },
                "About": "",
                "ImageField": "image",
                "ImagePrefix": "",
                "VideoPrefix": "",
                "DepthImagePrefix": "",
                "SourceImagePrefix": "",
                "MaxDistance": "300",
                "DEMPrefix": "",
                "Credentials": {
                        "Username": "",
                        "Password": ""
                },
                "Variables": {},
                "Filters": {},
                "Copyright": {
                        "text": "",
                        "url": ""
                },
                "GeoJSONFeatures": {},
                "GeoJSONCoverage": {},
        }
}
```


ServiceURL : The ServiceURL property in the OIC JSON points to a feature service. Its value will either be a feature service URL or an ArcGIS Online or Enterprise item URL or "**GeoJSONFeatures**". If the value of ServiceURL is GeoJSONFeatures, then the image data has been provided in the form of GeoJSON in the OIC instead of a feature service. It works as an identifier for oriented imagery tools to look for GeoJSONFeatures property in the OIC JSON to get the image metadata.

OverviewURL : The OverviewURL property in the OIC JSON points to a coverage map. Its value can either be a tile service URL or "**GeoJSONCoverage**". If the value of OverviewURL is GeoJSONCoverage, then the overview has been provided in the form of GeoJSON in the OIC. It works as an identifier for

oriented imagery tools to look for the GeoJSONCoverage property in the OIC JSON and create an overview from it.

GeoJSONFeatures: A property in the OIC JSON to store the **point GeoJSON** with properties similar to the ones defined in a feature class. Note: It is not a string but a JSON object.


```
{
    "type": "FeatureCollection",
    "crs": {
        "type": "name",
        "properties": {
            "name": "EPSG:4326"
        }
    },
    "features": [
        {
            "type": "Feature",
            "id": 7481,
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -84.66228658018147,
                    38.97238507020274
                ]
            },
            "properties": {
                "OBJECTID": 7481,
                "Name": "200614_201121890.jpg",
                "Image": "http://orientedimagerysamples.s3.amazonaws.com/200614_201121890.jpg",
                "CamHeading": 314.7228,
                "CamPitch": 91.12087,
                "CamRoll": -1.711208,
                "HFOV": 360,
                "VFOV": 180,
                "AvgHtAG": 4.5,
                "FarDist": 50,
                "NearDist": 1,
                "OIType": "B"
            }
        }
    ]
}
```

GeoJSONCoverage – A property in the OIC JSON to store the **polygon GeoJSON**. Note: It is not a string but a JSON object.


```
{
        "type": "FeatureCollection",
        "crs": {
            "type": "name",
            "properties": {
                "name": "EPSG:4326"
            }
        },
        "features": [
            {
                "type": "Feature",
                "geometry": {
                    "type": "Polygon",
                    "coordinates": [
                        [
                            [
                                -117.19683120946111,
                                34.05842899248937,
                                0
                            ],
                            [
                                -117.19749490473822,
                                34.058138419719409,
                                0
                            ],
                            [
                                -117.19743880793389,
                                34.058681710270047,
                                0
                            ],
                            [
                                -117.19740346092238,
                                34.05869701495445,
                                0
                            ],
                            [
                                -117.19683120946111,
                                34.05842899248937,
                                0
                            ]
                        ]
                    ]
                },
                "properties": {}
            }
        ]
    }
```


Limitation: It is required that the GeoJSON be in WGS84 system because the GeoJSONLayer object in the ArcGIS API for JavaScript 4.x only supports GeoJSON with WGS84. It does not support any other coordinate system.

Note: GeoJSONFeatures and GeoJSONCoverage are independent of each other. For example, there can be an OIC which references a point feature service for image metadata and has overviews defined in polygon GeoJSON format. Also, the OIC can have a point GeoJSON for exposure points and reference a vector tile layer for overviews.

GeoJSON format

The standard GeoJSON format is being followed, more information regarding the same can be found in the links provided below.

1. https://geojson.org/
2. https://en.wikipedia.org/wiki/GeoJSON
3. https://datatracker.ietf.org/doc/html/rfc7946
4. https://doc.arcgis.com/en/arcgis-online/reference/geojson.htm#:~:text=GeoJSON%20is%20an%20open%20standard,variety%20of%20geographic%20data%20structures.

Appendix C: Storing Imagery Geometry with FeatureService/FeatureClass/Tables

This section defines the schema for how such pixel geometry is defined such that the location of object recorded in image space as well ground space are recorded as well as allow one to be computed from the other.

It is often necessary to be able to store pixel space geometry as part of feature services, feature classes and tables. Typically, a user may digitize an object (eg fire hydrant) in an image and want to save the geometry of the object in pixel space as well as the geometry of the object in ground space. This then allows the location of the object to be shown and queried on a map and when selected display the accurate pixel geometry in the image. Alternatively, a user may digitize (or extract from deep learning) the pixel space geometry of an object in pixel space that is saved as a table. Then later want to compute the ground space geometry stored as a feature class or feature service. This transform from ground to image may occur at a later time when the orientation of the image has been better defined or when a more suitable model to define the 3rd dimension such as a depth image is available. In this section we will use the term Feature Service to mean feature service, feature class or table.

The feature services need to define two additional fields
ImgUrn – Image Universal Reference Number – This field provides a unique link back to the image stored in an OIC. Note that the name of the OIC is not defined and there may not necessarily be a 1:1 relation between the feature service and OIC. It may be necessary to search though more than one OIC to identify the correct image. This is by design. Based on a defined ImgUrn the application needs to find the image from the applicable OICs. An OIC references an Exposure feature service (EFS) that defines the location and orientation of the camera and includes a field called 'Image' that references the image. Note that multiple OICs can also reference the same EFS.
ImgGeom – Image Geometry defines the geometry of the object in pixel space. This can define a Point, Line, Polygon or Bounding Box.
The feature services may have any other fields. Typically, such a feature services would have fields such as 'Category' etc. that define what the type of object is e.g. Car, Lamp Pole, etc.

ImgUrn

Image Universal Reference Number
Unique ID to identify the original image in which the object was recorded / labeled.
This ImgUrn is constructed as follows:
If the EFS in the OIC has a 'GlobalID' field
    ImgUrn = GlobalID of the record in the EFS
Else if EFS in the OIC has a 'ExposureID' field
    ImgUrn = ExposureID of the record in EFS
Else
    If EFS in the OIC has 'Name' field
        ImgURN = Name+"|"+EFS name+"|"+OBJECTID
    Else
        ImgURN = "|"+EFS name +"|"+OBJECTID
    ImgURN = first 50 characters in the above string

ImgGeom

Stores the pixel geometry of the object or label as a JSON
X is columns and Y is rows, with the upper left pixel being 0,0. Such image coordinates are typically defined as integers. If using floating point, then the coordinate is relative to the center of the upper left pixels. Note that coordinates beyond the image extent are allowed in some cases but are typically extrapolated locations.

For Point Image Geometry

{"x": 100, "y":200} or
{"x": 200, "y":"300", "face": "F"} for cube images with six faces. Possible face values are F, R, B, L ,D, U

For Bounding Box Image Geometry

{"xmin": 300, "ymin": 200, "xmax": 500, "ymax": 300, "pos": "BC"} or
{"xmin": [300,"F"], "ymin": [200,"R"], "xmax": [500,"R"], "ymax": [300,"F"], "pos": "BC" } for cube images
Possible "pos" values are "BC", "LC", "TC", "RC", "CC"
If undefined pos = CC
A bounding box is defined as a point feature, but the location of that point in the map is dependent on how the feature was drawn. When looking at a nadir image the location of the object is equivalent to the center of the bounding box. When the bounding box represents a vertical object (eg lap pole) then the location of the feature on the ground is defined by the lower center of the bounding box i.e. "LC". The "BC" (bottom center), "TC" (top Center) ,"RC" (right center) are define to enable support for image that are rotated. If the image was rotated such that upper left point of the image is at the top right of the display (when looking at objects with top up) then then the bounding box should be labeled "RC"

For Polygon Image Geometry

{"rings":[[[2832.78,1057.24],[2913.55,2724.9],[1570.57,2789.55],[1282.78,1034.19],[2832.78,1057.24]]]
}
For cube images,
{"rings":[[[2832.78,1057.24,
"D"],[2913.55,2724.9,"D"],[1570.57,2789.55,"F"],[1282.78,1034.19,"F"],[2832.78,1057.24,"D"]]]}

For Polyline Image Geometry

{"paths":[[[3084.05,1240.65],[3070.72,1561.82]]]}

For cube images,

{"paths":[[[3084.05,1240.65,”F”],[3070.72,1561.82,”R”]]]}

Note:- Cube images are different from bubble / equirectangular images. Cube here means that there is different image for each face of the cube.