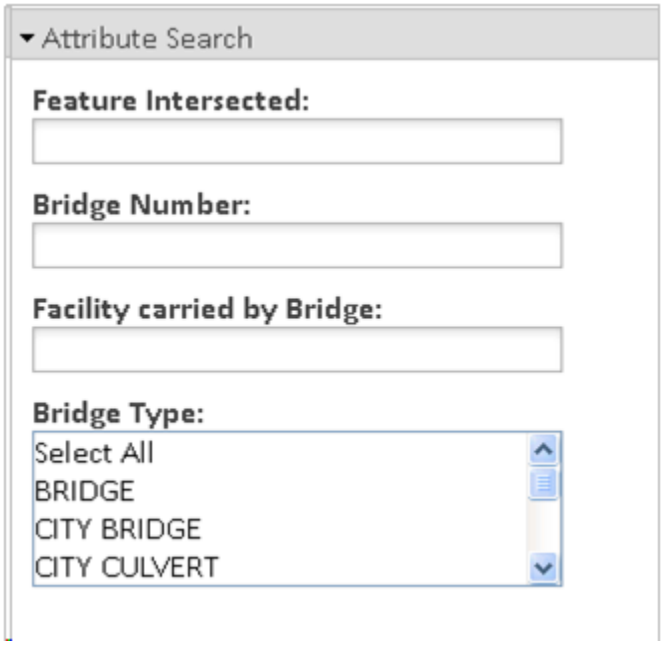
Environmental Review Tool Search Configuration JSON Specification

By default, the Attribute Search section of the Feature Search tool utilizes text searches of all visible fields. Given that the data within the layer is not readily exposed, this makes successfully performing an attribute search difficult. The following screenshot shows an example of a layer which contains 3 attributes which are searchable followed by the fourth, Bridge Type, which utilizes a domain list. The Feature Search tool can be configured to limit the fields displayed within the Attribute Search section of the Feature Search tool as well as configure the use of dropdown lists for specified fields.



There are two ways to limit the fields/attributes listed in the Attribute Search section of the Feature Search tool:

- Indicate which fields should be visible within the ArcGIS map document (.mxd) prior to publishing (see the Field Aliases & visibility section of the Map Service Specifications for details), or
- Configure the Feature Search tool by writing JSON code to include only those fields for display and enter the code in the Find JSON section of the Layer.

Additionally, there are two ways to define the values within the dropdown list:

- Create domain table(s) for the attribute(s) and include the domain table in the ArcGIS map document (.mxd) prior to publishing (see the Domain Tables section of the Map Service Specifications for details), or
- Manually define the values within the JSON code and enter the code in the Find JSON section of the Layer.

Writing the JSON code to configure the Feature Search

Javascript Object Notation (JSON) and can be thought of as groupings of objects with name/value pairs as <name>:<value> in between each matching bracket {}. Arrays of these objects are delimited by square brackets []. It starts with an array of layers in the published map resource, matching the indexes shown within the details of the service (see View Details in Layers help topic).

As can be seen in the example below, all contain a readable name (which should be initialized from the aliases published to the mapservice - see the Field Aliases & visibility section of the Map Service Specifications for details regarding defining aliases), and an attr, which points to the actual attribute being referenced. The attr object is an array of the attributes in the table and how they are treated. By default, that's all that's needed and it will assume the attribute is a string match.

Tips for writing successful JSON code

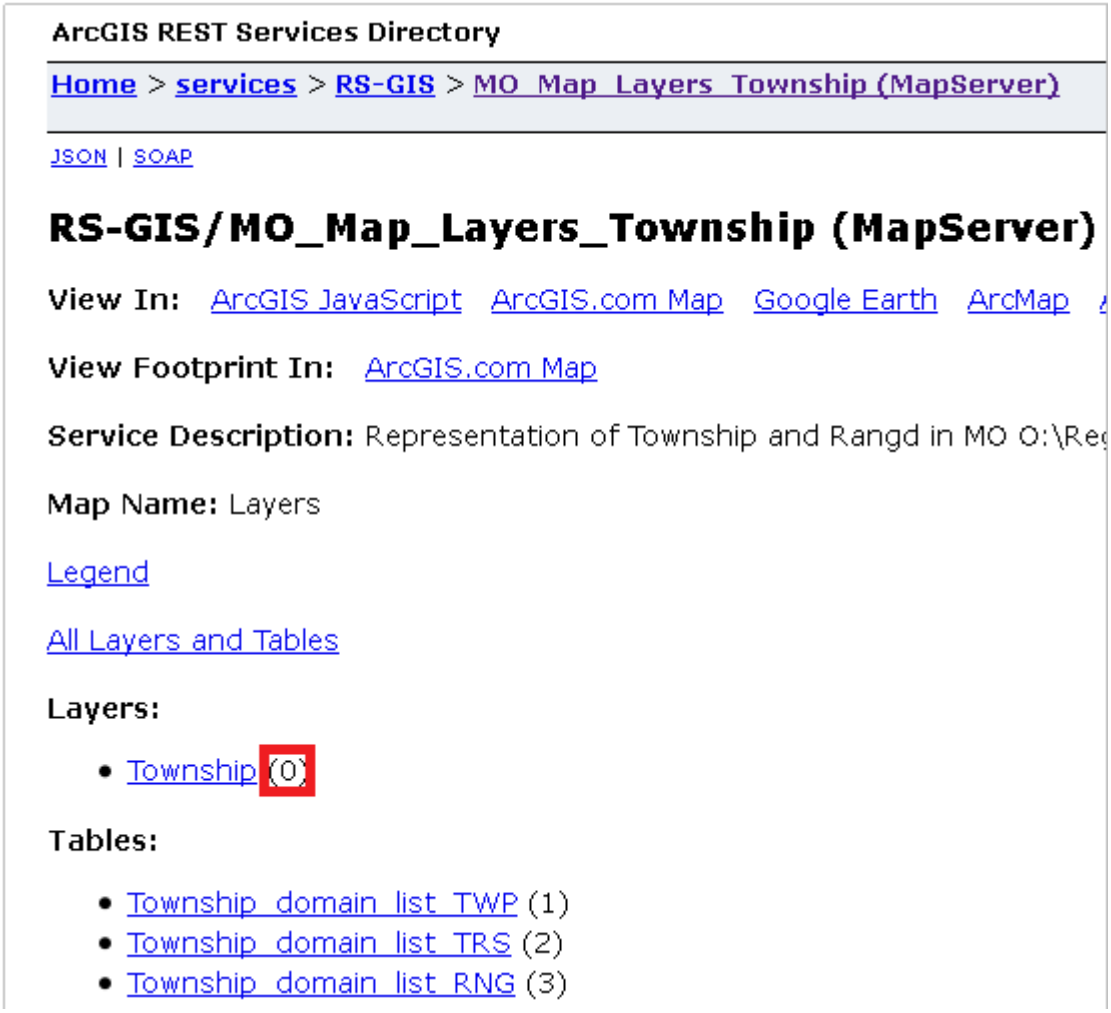
- Copy the JSON code from a Layer which already has the Feature Search successfully configured and edit it for use with the new layer - that way you're not creating the code from scratch but utilizing a working example.
- Utilize Notepad++ (freeware downloadable from the internet) or another text editor which helps to identify matching brackets, parenthesis, etc. Dangling or mismatched brackets, parenthesis, etc. will cause the JSON code to fail.
- After saving the JSON code within the Layer, immediately test the Attribute Search section of the Feature Search tool for the configured layer to verify that it is working successfully. If the Map is already open, refresh the page prior to testing.

JSON code components defined

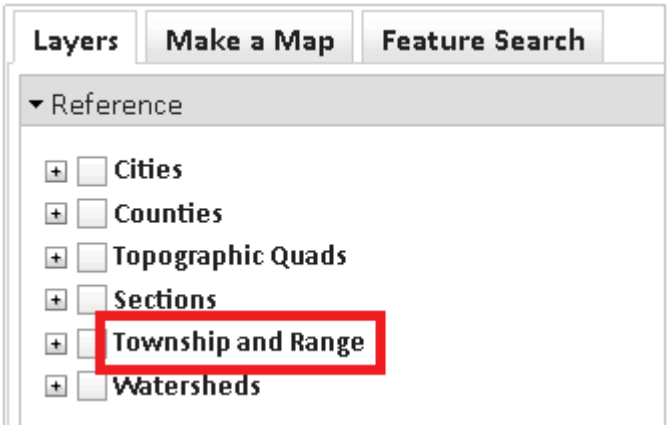
NOTE: the screenshots below do not match the JSON Example code.

- **name**: name of Layer within the map service

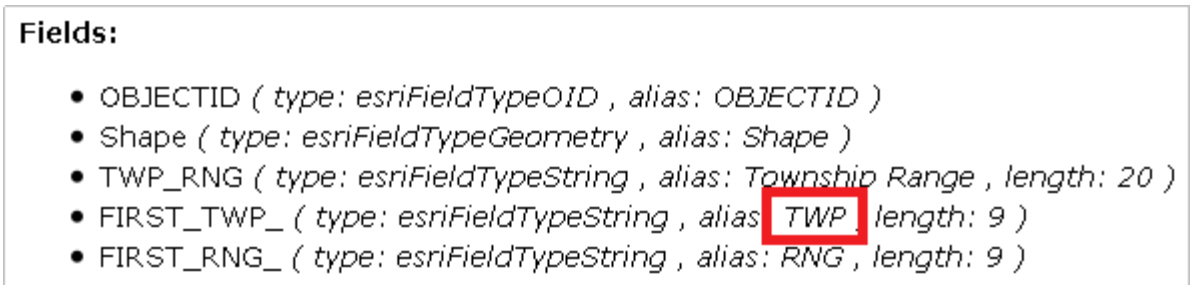
- **index**: index of the layer/sub layer as defined within the map service detail



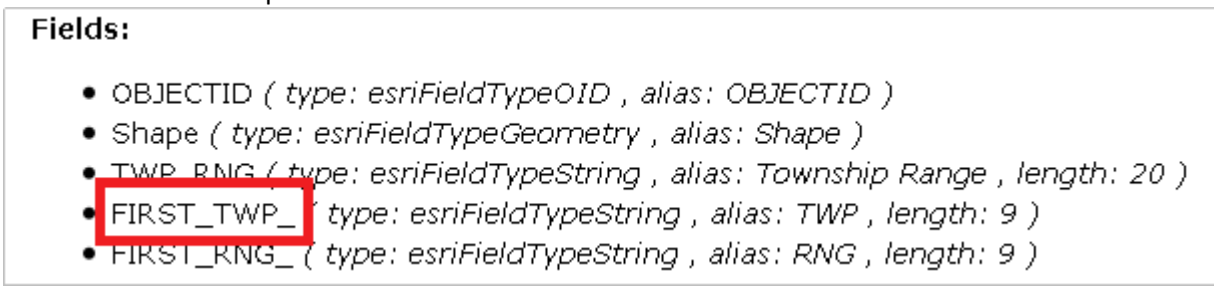
- **parent**: name of layer within ERT Map's table of contents, defined as the Layer Title within Layers.



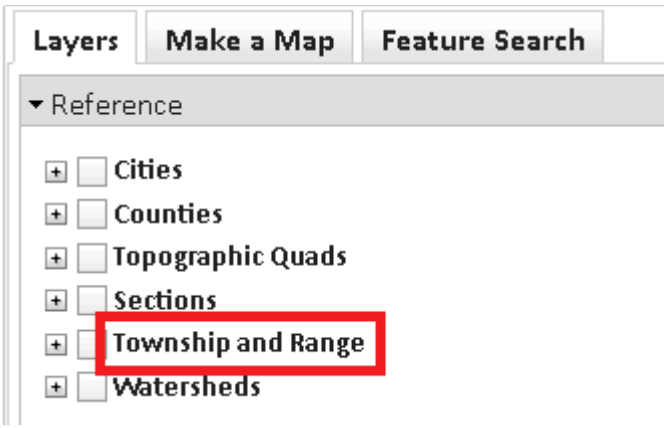
- **name** (within attrs section): alias of attribute as defined within the Fields section of the Layer within the map service



- **attr** (within attrs section): name of attribute as defined within the Fields section of the Layer within the map service



- **type** (within attrs section): indicates search types other than string match.
    - **noFind** - not findable,
    - **number** - numeric range,
    - **comboAttr** - allows two or more attributes to be grouped such that they can be combined differently than the default AND,
    - **select** - dropdown list,
    - **selectType** (within select section): indicates whether the dropdown list will be manually defined (data) or utilize a domain list within the map service (dataquery) or use the selection of one dropdown list to dynamically populate a second dropdown menu (subquery)
        - **data**: used to manually define values within dropdown list values
            - **size** (OPTIONAL): determines the size of the selection list
            - **exact** (OPTIONAL): makes the comparison = instead of like
            - **val**: data value as it appears within the layer's attribute table
            - **disp**: value as it should actually be displayed, as the val is often a code or not user-friendly.
        - **dataquery**: use a domain table published within the map service to define the values within the dropdown list
            - **queryLayer**: same as parent - name of layer within ERT Map's table of contents, defined as the Layer Title within Layers.



- **querySubLayer**: index of the domain table (layer/sub layer) as defined within the map service detail

- **querySubLayerName**: name of domain table (table/layer) within the map service

- **outFields**: name of attribute within domain table as defined in the map service

Layer: Township_domain_list_TWP (ID: 1)

Name: Township_domain_list_TWP

Display Field: FIRST_TWP_

Type: Table

Geometry Type: N/A

Description: null

Definition Expression: N/A

Copyright Text: N/A

Default Visibility: false

MaxRecordCount: 1000

Supported Query Formats: JSON, AMF

Supports Advanced Queries: true

Supports Statistics: true

Use Standardized Queries: true

Extent:

Drawing Info:

N/A

HasZ: false

HasM: false

Has Attachments: false

HTML Popup Type: esriServerHTMLPopupTypeNone

Type ID Field: null

Fields:

- OBJECTID *(type: esriFieldTypeOID , alias: OBJECTID)*
- FREQUENCY *(type: esriFieldTypeInteger , alias: FREQUENCY)*
- FIRST_TWP_ *(type: esriFieldTypeString , alias: TWP , length: 9)*

- **valField**: name of field/attribute which contains the display/value attributes to populate the list from a map service query - typically the same as outFields
- **dispField**: name of field/attribute which contains the display/value attributes to populate the list from a map service query - typically the same as outFields

- **subquery**: same as dataquery but allows specifying another feature search attribute (queryField) as a name of another field in this layer which when values are selected, a query using only those values will be done against the sub-table's specified attribute (queryAttr) in order to dynamically populate the field, such as only populating the sub-watersheds intersecting a selected watershed field.

NOTE: For map services which have layers within Groups, the group is reflected in the JSON with {unused: true} as the Feature Search tool only applies to feature classes, not groups. So, in the following map service example in which there are 3 Groups at Index 0, 4, and 8, this is reflected by {unused: true} within the JSON 3 times, whereas the feature classes are reflected according to the instructions above. Expand this section to view the example.

Layers:

- Tier 1 At-risk Species (0)
  - Tier 1 Outlines (1)
  - Tier 1 Shapes (2)
  - Tier 1 Centroids (3)
- Tier 2 At-risk Species (4)
  - Tier 2 Outlines (5)
  - Tier 2 Shapes (6)
  - Tier 2 Centroids (7)
- Natural Communities (8)
  - Community Outlines (9)
  - Community Shapes (10)
  - Community Centroids (11)

Tables:

- tier1_sname_domain (12)
- community_scomname_domain (13)
- tier2_sname_domain (14)
- tier2_scomname_domain (15)
- tier1_scomname_domain (16)

the JSON is as follows:

```
        queryLayer: "Element Occurrences (NENHP)",
        querySubLayer: 12,
        querySubLayerName: "tier1_sname_domain",
        outFields: ["SNAME"],
        valField: "SNAME",
        dispField: "SNAME"
      }
  }, {
    name: "Common Name",
    attr: "SCOMNAME",
    type: "select",
      select: {
        selectType: "dataquery",
        queryLayer: "Element Occurrences (NENHP)",
        querySubLayer: 16,
        querySubLayerName: "tier1_scomname_domain",
        outFields: ["SCOMNAME"],
        valField: "SCOMNAME",
        dispField: "SCOMNAME"
      }
  }]
},
{
  unused: true
},
{
  name: "Tier 2 Outlines",
  index: 5,
  noTableBorder: false,
  parent: "Element Occurrences (NENHP)",
```

.....

JSON Example

The example text can be downloaded here.

```
            {val: "LE", disp: "LE - Listed endangered"},
            {val: "LT", disp: "LT - Listed threatened"}]
        }
      }]
    }, {
      name: "County",
      attr: "CONAME",
      type: "select",
      select: {
        selectType: "dataquery",
        queryLayer: "Counties",
        querySubLayer: 0,
        querySubLayerName: "Counties",
        outFields: ["NAME"],
        valField: "NAME",
        dispField: "NAME"
      }
    }, {
      name: "Watershed",
      attr: "WSHED_CD",
      type: "select",
      select: {
        selectType: "data",
        data: [
          {val: "SELECTALL", disp: "Select All"},
          {val: "02040303", disp: "02040303 - Chincoteague"},
          {val: "02040304", disp: "02040304 - Eastern Lower Delmarva"},
          {val: "02070001", disp: "02070001 - So. Branch Potomac River"},
          {val: "02070003", disp: "02070003 - Cacapon - Town"}]
      }
    }, {
      name: "Subwatershed",
      attr: "SHU_CD",
      type: "select",
      select: {
        selectType: "subquery",
        queryField: "Watershed",
        queryLayer: "NHDE Resource Search",
        querySubLayer: 1,
        querySubLayerName: "shuclink",
        queryAttr: "wshedcd",
        outFields: ["*"],
        valField: "Shu_cd",
        dispField: "Shu_name"
      }
    }]
  }]
```