Initial Question / Scope

- The scope of the PDS Search API we are developing is poorly defined.
- What are people allowed to be searching on? Bundles/collections or products? The answer will affect how the query and response parameters are defined.
    - Some of us are always thinking about product-level search
    - We agreed at June meeting to constrain initial drafts to collection-level search

Guiding Principles

- **REPRODUCIBILITY:** The response has the capability to reconstruct the query that was made, regardless if the query is for a collection or product.
    - The response object should include the request parameters by default, including the endpoint that was targeted.
        - A user might target Image Atlas with a product-level query, then Atlas has to do some processing in the background to handle that request. But ODE might handle the same request differently.
- **MINIMAL BY DESIGN:** The response should include the minimal amount of metadata required to respond to a query. A default set should be defined that could be extended to include additional fields.
    - Should there be a default set? Or should only the exact fields that were requested be returned?
    - Could include ways to extend result fields in request (verbose mode).
        - A user could provide a specific list of fields they want returned in response
        - Could include predefined levels of verbosity
            - These sets could be challenging to define, as data sets are not necessarily consistent with one another across the PDS.
- **WELL DOCUMENTED RESPONSE:** The response should include pointers to additional information about the search process and search results. The documentation should explain assumptions, caveats, and limitations or the search service providing the response.
- **WELL DOCUMENTED ERRORS:** Responses should include support for special error/warning cases to help end-users understand *why* given queries did not return expected results. Errors/warnings can also help improve the search capability itself by learning which queries are failing most often. At a minimum, the following could be supported:
    - Field exists and does not match the data
    - Field does not exist

Questions/Discussion

- As a starting point, we encourage the group to think about developing a product-level search that is restrictive about which fields you can search on
  - A product-level search on a handful of bundles/datasets, also time, instrument, location, 5 fields or something, that will get you a pretty good variety of different cases with different combinations. Will also get you thinking about things like do I need row numbers/paging.
    - Ex; these are the columns that I want back in my table.
- What are we actually responding to?
  - Does a collection-level search only return information about collections? Does collection information include descendent data products?
- The API documentation should include definitions for how the API will format different data types:
  - Text fields
  - Numerical fields
  - Lat/long objects <= how complex is this? How would these be stored in the label?
  - etc.
  - We should aim to be consistent across all response formats.

Notes

- Image Atlas is using something called "views"
  - Pre-created views of the data that don't exactly exist in the data model
  - Views by mission, orbital mission, etc.
    - Aggregates data together in useful ways
  - Further down the line- some kind of aggregation
    - Some sort of search across *missions*, for example
    - Atlas can do this, but only for images
  - We could either stick strictly to what PDS4 data model provides for a mission; if you want to do multiple searches across multiple missions, a user has to do multiple queries. Or, we could aggregate these results.
- There's a lot more to search than just going through PDS4 labels. Often the data model has to be massaged to get good results.
  - Is a user searching for a "poster-child" image, or the other 7,000 derived products?
  - For a query like: "sol 319; orbital images on Mars" (construct a query where you are looking at multiple missions/instruments but not images particularly, but all data related whether it's images, raw data, instrument data, cartography, etc.) right now that's not something image node can handle because it only handles images.

    - There’s not really any way to do a flexible response structure that includes more than the PDS4 structure as it exists: here is the information in a different view.
      - Do we want to include in our responses different layers of abstraction?
        - Here is a response that includes images, cartography, etc. all in one.
    - A real challenge for user in this example;
      - A query goes out to different nodes might get different responses back
      - How will a user know that they will get different responses from a single query that is sent to different nodes?
      - We can do that, but a user has to know that.
      - Talking to the users will help us answer this.
  - Maybe a ZIP response format that includes info about instrument/images/cartographic data being returned. Maybe it’s three responses all zipped up.
    - Alternatively, users could just have to make three different requests. People are going to have to know something about what info they’re after.
      - For example, a user will not go to pds.gov for CTX data- they will target Atlas API for specific data sets.
        - Is it like that most people will take this approach?
        - Most users will have to understand the data/results they’re after.
    - We should start simpler and expand in the future.
  - For the purpose of broader scope integration, specifically with search engines, PDS API Response Formats should follow a known schema at schema.org/docs/schemas.html or be registered with schema.org
- Aggregation in responses to support faceted queries: because of significant data gaps within the heterogeneous data sets between PDS nodes, it’ll help future user experiences if we support the ability to return counts of:
  - 1) which fields are available within given data sets being queried
  - 2) the counts of the possible values for given fields available