PDS APIs

Release B15.0

NASA Planetary Data System

Nov 13, 2024

CONTENTS

The Planetary Data System (PDS) is a federated system of nodes that archive planetary science data.

The PDS Application Programming Interface (API) provides a consistent way for planetary science community to discover and share archival data across PDS. This API is one of the cornerstone applications for providing an integrated worldwide data services platform that enables the efficient discovery, dissemination, use and analysis of internationally sponsored planetary science archives.

PDS is willing to develop ReST-ful web APIs for different applications (so far, registry, dois).

These pages document how to access these APIs.

This web site is also available as a PDF document

CHAPTER

ONE

OVERVIEW

The PDS API base urls are provided under the following pattern:

```
https://pds.nasa.gov/api/{service}/{version}/{service_path+params}
```

where:

- `{service}`: the service such as 'search' (in registry), 'doi', etc..
- `{version}`: the version of the service.
- `{service_path+params}`: the ReST path for the service, including any query parameters - this is essentially the remaining portion of the URI after the version.

API entries currently available are:

service	version	scope	specification	user's guide
search	1.5	search PDS data archive	search_spec	search_guide
doi	0.2	manage PDS DOIs	doi_spec	

So for example:

```
https://pds.nasa.gov/api/search-geo/1/products?limit=10
```

intends to obtain 10 product entries from the 1.1 version of the GEO node's search (registry).

The API specifications design is driven by the *PDS API general conventions*

CHAPTER

TWO

SEARCH API USER GUIDE

Note

The current guide is based on the PDS Search API version 1.1

Warning

Since our servers are not fully populated with all PDS data sets, the examples presented in this user guide may return empty results or 404 (Not Found) errors. If there is a data set you would like added, please contact the PDS Help Desk[1] for assistance.

The PDS Search API provides endpoints:

- to **search** for bundles, collections and any PDS products with advanced search queries.
- to **browse** the archive hierarchically downward (e.g. collection's products) or upward (e.g. bundles containing products),
- to **resolve** an identifier (lid or lidvid) and retrieve the product label and data where ever it is in the Planetary Data System.

For an introduction to the Search API and what it's capable of, checkout these slides and a presentation given at the 2022 PSIDA conference:

- Slides: https://www.cosmos.esa.int/documents/6109777/9316710/X03+-+Padams+-+NASA_Planetary_Data_System_Search_API.pdf
- Presentation Recording: https://www.cosmos.esa.int/web/psida-2022/-/x03

These pages provide a user guide for the PDS Search API.

2.1 Quickstart

The following section provides a quickstart guide to try out the PDS Search API.

Warning

[1] pds-operator@jpl.nasa.gov

Since our servers are not fully populated with all PDS data sets, the examples presented in this user guide may return empty results or 404 (Not Found) errors. If there is a data set you would like added, please contact the PDS Help Desk[2] for assistance.

Note

curl command line tool is used to request the API in this documentation. curl is available in many operating systems by default. If not, you can get curl from https://curl.se/ or using a package management tool specific to your operating system (brew, apt, ...).

2.1.1 Search With curl

1. Open a Terminal window (or your favorite command-line application).
2. Get 5 products' metadata from the API in JSON format:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products?limit=5' \
    --header 'Accept: application/json'
```

3. Get 5 products' metadata from the API in XML format:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products?limit=5' \
    --header 'Accept: application/xml'
```

4. To view this in a more readable way, you can pipe the output to a file, or pretty print (on Mac/Unix):

```
# Output JSON to a File
curl ... > my_first_query.json

# Pretty print JSON
curl ... | json_pp > my_first_query.json

# Output XML to a File
curl ... > my_first_query.xml
```

More details on how to use the API can be found in the *endpoints* .

2.1.2 Search with Python

Alternatively, it is possible to use other tools such as Postman and programming languages such as Python to call the PDS Search API.

To use the **PDS API Python Client**, you can read this other Quickstart[3]

2.2 Query Syntax

Warning

[2] pds-operator@jpl.nasa.gov
[3] https://nasa-pds.github.io/pds-api-client/quickstart

Since our servers are not fully populated with all PDS data sets, the examples presented in this user guide may return empty results or 404 (Not Found) errors. If there is a data set you would like added, please contact the PDS Help Desk[4] for assistance.

Note

curl command line tool is used to request the API in this documentation. curl is available in many operating systems by default. If not, you can get curl from https://curl.se/ or using a package management tool specific to your operating system (brew, apt, ...).

2.2.1 Endpoints

The URLs for performing GET requests for searching PDS data are as follows.

The **base URL** of the PDS Search API, for search across all the PDS nodes, is:

```
https://pds.nasa.gov/api/search/1/
```

The main use cases, to search, crawl products or resolve a product identifier are given in the following sections.

2.2.2 Search Products

Request Examples

Get the list of properties which describe the products, which criteria you can search against:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/properties'
```

Search for products which processing level is "Raw", using the property "pds:Primary_Result_Summary.pds:processing_level" found before, get 10 results:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products' \
    --data-urlencode 'limit=10' \
    --data-urlencode 'q=(pds:Primary_Result_Summary.pds:processing_level eq "Raw")'
```

Request Template

The requests template is a follow:

```
GET /api/search/1/classes/{product_class}[?[{query-parameter}={query-parameter-value}]*]␣
↪HTTP/1.1
Host: pds.nasa.gov
```

The list of *product_class* proposed by the API can also be found from URL:

```
https://pds.nasa.gov/api/search/1/classes
```

The concept of product class is derived from the PDS4 standard[5].

[4] pds-operator@jpl.nasa.gov
[5] https://pds.nasa.gov/datastandards/documents/im/current/index_1I00.html

Query Detailed Syntax

Query Parameters

The query parameters are:

Query Parameter	Description	Example
q	(Optional, string) Query string you wish to parse and use for search. See *query string syntax*	q=target_name eq "Mars"
keywords	(Optional, string) String used for text search on title and description of the PDS4 labels	keyword=insight
fields	(Optional, array of strings) Array of fields you wish to return.	fields=pds:Time_Coordinates.pds:start_da
sort	(Optional, array of strings) Array of fields you wish to sort by, mandatory when search-after is used	sort=ops:Harvest_Info.ops:harvest_date_ti
search-after	(Optional, string or number) For pagination, the page will start from the value of the field selected in *sort*.	search-after=2024-01-23T22:53:30.402453Z
limit	(Optional, integer, default=100) The number of records/results to return. By specifying a value of 0 only the summary of the results is returned, not the individual results.	limit=100

q and *fields* use PDS4 *Fields Dot Notation*

Query String Syntax

An example of query syntax (*q* query parameter) is:

For example:

```
((pds:Primary_Result_Summary.pds:processing_level eq "Raw") and not (ops:Data_File_Info.
↪ops:file_size ge 8942))
```

The query syntax follows the rules:

```
{query} = {comparison}|{group}

{comparison} = {field} {comparison operator} {literal value}

{group} = [not] ({comparison} [[and|or] {group}])
```

- **{field}** follows the *Fields Dot Notation* . The available fields can be found in responses *summary* object, *properties* attribute.
- **{comparison operator}** are eq, ne, gt, lt, ge, le
- **{literal value}** is either a string between " (double quotes) or a numerical value (float or integer).
- Wildcard searching is available with the **like** operator. The wildcarding syntax of the **{literal value}** follows the [OpenSearch Simple Query String](https://opensearch.org/docs/latest/opensearch/query-dsl/full-text/#simple-query-string) convention.
- **{group}** has mandatory parentheses (round brackets) which make any complex query loaded with parentheses, as seen in the example above, don't forget them !

Warning

the `like` operator does not work because of a known bug[6]

Operator	Description	Example
Comparison Operators		
eq	Equal	target_name **eq** "Mars"
like	Similar to	target_name **like** "mars"
ne	Not equal	target_name **ne** "Saturn"
gt	Greater than	pds:Time_Coordinates.pds:start_date_time **gt** "2001-05-10T00:00:00Z"
ge	Greater than or equal	pds:Time_Coordinates.pds:start_date_time **ge** "2001-05-10T00:00:00Z"
lt	Less than	pds:Time_Coordinates.pds:start_date_time **lt** "2020-06-01T00:00:00Z"
le	Less than or equal	pds:Time_Coordinates.pds:start_date_time **le** "2020-06-01T00:00:00Z"
Logical Operators		
and	Logical and	target_name **eq** "Mars" **and** instrument_name **eq** "hirise"
or	Logical or	target_name **eq** "Mars" **or** target_name **eq** "Phobos"
not	Logical negation	**not** target_name **eq** "Mars"
Grouping Operators		
( )	Precedence grouping	((target_name **eq** "Mars" **or** target_name **eq** "Phobos") **and** (instrument_name **eq** "hirise"))

Fields Dot Notation

General Case

The syntax of the field names use a combination of the PDS4 Information Model and dot notation[7] representations of an XML XPaths.

Query parameters will use a combination of an attribute with its parent class in *all lowercase*:

```
{namespace:parent_class}.{namespace:attribute}
```

For example:

```
pds:Science_Facets.pds:discipline_name
pds:Investigation_Area.pds:type
```

The classes and attributes are defined in the PDS4 Data Dictionnaries[8].

The PDS4 data dictionaries are augmented with a specific *ops Namespace* which contains attributes managed by the PDS Registry[9] in addition to the PDS4 labels attributes.

NOT IMPLEMENTED

In the event that the {parent_class}.{attribute} combination does sufficiently guarantee uniqueness or sufficiency of

[6] https://github.com/NASA-PDS/registry-api/issues/170
[7] http://reeborg.ca/docs/oop_py_en/oop.html
[8] https://pds.nasa.gov/datastandards/dictionaries/index-1.18.0.0.shtml
[9] https://nasa-pds.github.io/registry/

search when a class is inherited by multiple classes, additional ancestor classes should be prepended to the query parameter until sufficient uniqueness is attained:

{ns:ancestor_class}.{ns:parent_class}.{ns:attribute}

If the query parameter grows beyond 3 ancestor classes, a :ref:custom query parameter <Custom Query Parameters> should be considered.

In the event that multiple attributes are to be grouped together for search, the parent class should be used as the query parameter:

{ancestor_class}.{parent_class}

Custom Query Parameters

NOT IMPLEMENTED

There are several cases where custom query parameters are preferred over the Dot Notation, but should only be avoided wherever possible in order to minimize confusion amongst developers attempting to use the API. These are also subject to approval by Search Integration Working Group representative for each node. That member is responsible for providing those updates to Engineering Node.

Some reasons for custom query parameters:

- Combination of multiple attribute values into one

- Special cases where XQuery needs to be used for finding specific values (e.g. instrument/spacecraft described in Observing_System_Component class)

- Custom search fields on non-PDS4 metadata (e.g. image tags, operations note, etc.)

- Support common search or PDS4 terminology (e.g. target_name, lidvid)

2.2.3 Resolve A Product Identifier

Default Resolution

If you know the lid (for example *urn:nasa:pds:insight_rad*) or lidvid (for example *urn:nasa:pds:insight_rad::2.1*) identifier of a product, you can retrieve its description, whereever it is managed in the PDS system, with the following request:

```
https://pds.nasa.gov/api/search/1/products/{identifier}
```

For example

```
curl --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad::2.1' \
    --header 'Accept: application/json'
```

Search for Latest vs. All Versions

Latest Version

By default, when the identifier is a lid (without a version, for example urn:nasa:pds:insight_rad) only the latest description of the product is returned.

The request:

```
https://pds.nasa.gov/api/search/1/products/{lid}
```

2.2.4 Crawl a Data Set Hierarchy

For a given product with identifier *lidvid1*, you can browse its parent products (member-of) or children (members).

Get the Collections of a Bundle

Get its **children** (collections):

```
https://pds.nasa.gov/api/search/1/products/lidvid1/members
```

For example, run:

```
curl --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad::2.1/
↪members' \
    --header 'Accept: application/json'
```

The same request can be used to get the observational products or documents of a collection from the collection's lidvid.

Get the Observational Products of a Bundle

```
https://pds.nasa.gov/api/search/1/products/lidvid1/members/members
```

For example, run:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad::2.1/
↪members/members' \
    --header 'Accept: application/json'
```

Get the Collection or Bundles of an Observational Product

Get its **parent** (collection):

```
https://pds.nasa.gov/api/search/1/products/lidvid1/member-of
```

The same request can be used to get the bundles of a collection from the collection's lidvid.

Get its **grandparent** (bundle):

```
https://pds.nasa.gov/api/search/1/products/lidvid1/member-of/member-of
```

For example, run:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad:data_
↪raw:hp3_rad_raw_00004_20181130_085325/member-of/member-of' \
    --header 'Accept: application/json'
```

2.2.5 Use pagination to get all products matching a request

When you're searching for a large number of products, you'll need to use pagination to ensure you receive all the results. Here's how you can do it.

> **Note**
>
> The pagination parameters (sort, limit, search-after) described in this section are applicable to all the end-points.

To start, let's say you want to get all the members of a collection named "OSIRIS-REX Spectrometer calibrated observations", which identifier is *urn:nasa:pds:orex.ovirs:data_calibrated::11.0*. You can use the following request in a web browser:

```
https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:orex.ovirs:data_calibrated::11.0/
↪members
```

This request will only give you the first 100 products out of the total available in the collection (in this example, 334,940 products).

To get all the results, you need to use the pagination.

1. Make the Initial Request:

Sort the results by the harvest time, which is the time when products were loaded into the registry. You can do this using the *curl* command:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:orex.ovirs:data_
↪calibrated::11.0/members'
   --header 'Accept: application/json'
   --data-urlencode 'sort=ops:Harvest_Info.ops:harvest_date_time'
```

You are getting the first 100 products, members of the collection, sorted by harvest time (time when they were loaded in the registry).

2. Get the Next Page

To retrieve the next set of results, you need to get the latest harvest date and time from the previous response. This information is included in the description of the last product returned.

```
"ops:Harvest_Info.ops:harvest_date_time": [
   "2023-05-26T05:53:24.611495Z"
],
```

Use this latest harvest date and time as the reference for the next request:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:orex.ovirs:data_
↪calibrated::11.0/members'
   --header 'Accept: application/json'
   --data-urlencode 'sort=ops:Harvest_Info.ops:harvest_date_time'
   --data-urlencode 'search-after=2023-05-26T05:53:24.611495Z'
```

3. Iterate Until Completion:

Keep making requests and updating the *search-after* parameter with the latest harvest date and time until the number of products returned is less than the limit (100 in this case).

4. Changing Pagination Parameters:

You can adjust the default limit of 100 products per page using the limit parameter. For example:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:orex.ovirs:data_
↪calibrated::11.0/members'
   --header 'Accept: application/json'
   --data-urlencode 'limit=500'
   --data-urlencode 'sort=ops:Harvest_Info.ops:harvest_date_time'
   --data-urlencode 'search-after=2023-05-26T05:53:24.611495Z'
```

Note

The number of products per page, specified by the limit parameter should be below a few 1000s. If the queries are too demanding for the server, you might experience 504 errors[10] . The optimal number of product per page for quicker data retrieval depends on the type of response requested, as shown in statistics on pagination performances[11].

2.3 Responses

Note

curl command line tool is used to request the API in this documentation. curl is available in many operating systems by default. If not, you can get curl from https://curl.se/ or using a package management tool specific to your operating system (brew, apt, ...).

2.3.1 Content Negotiation, Formats

Principles

A simple style of content negotiation[12] is used to match the format requested by the client and the capability of the server.

The client can specify the desired response format by including the HTTP header `Accept`. If no `Accept` header is present in the request, or if the requested content type is not available, the server will provide the response in JSON format by default.

The following table provides a list of the supported HTTP Accept header types:

Accept Header	Format	Note
application/json	JSON	Simplified JSON view of the PDS4 metadata label. Contains "flattened" PDS4 properties extracted from the metadata label
application/xml	XML	Same as application/json, but in an XML
application/vnd.nasa.pds.pd	JSON	JSON response containing the full PDS4 metadata translated to JSON, along with some additional supplemental
application/vnd.nasa.pds.pd	XML	Same as application/vnd.nasa.pds.pds4+json, but in an XML format. This response format contains the original PDS4 labels.
application/kvp+json	JSON	JSON response containing key-value-pairs for the applicable metadata.
text/csv	CSV	Returns a CSV table containing values for the parameters in the request. If no parameters were specified in the request, a default set is returned. The first row of the CSV is a header that describes the values in each column.
text/html	HTM	JSON response embedded in an HTML body. This format is provided for requests coming from the browers (e.g. Google Chrome) URL bar.

`application/vnd.nasa.pds.pds4+json` and `application/vnd.nasa.pds.pds4+xml` have been chosen to comply with RFC6838[13]

[10] https://github.com/NASA-PDS/registry-api/discussions/521
[11] https://github.com/NASA-PDS/registry-api/issues/552#issuecomment-2389199054
[12] https://restfulapi.net/content-negotiation/
[13] https://datatracker.ietf.org/doc/html/rfc6838

Examples

application/json

The request:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad::2.1
↪' \
    --header 'Accept: application/json'
```

Returns


```
{
    "id": "urn:nasa:pds:insight_rad::2.1",
    "type": "Product_Bundle",
    "title": "Mars InSight Lander Radiometer Data Archive",
    "start_date_time": "2018-05-05T00:00:00Z",
    "stop_date_time": "3000-01-01T00:00:00.000Z",
    "investigations": [
        {
            "id": "urn:nasa:pds:context:investigation:mission.insight",
            "href": "http://localhost:8080/products/
↪urn:nasa:pds:context:investigation:mission.insight"
        }
    ],
    "observing_system_components": [
        {
            "id": "urn:nasa:pds:context:instrument_host:spacecraft.insight",
            "href": "http://localhost:8080/products/urn:nasa:pds:context:instrument_
↪host:spacecraft.insight"
        },
        {
            "id": "urn:nasa:pds:context:instrument:radiometer.insight",
            "href": "http://localhost:8080/products/
↪urn:nasa:pds:context:instrument:radiometer.insight"
        }
    ],
    "targets": [
        {
            "id": "urn:nasa:pds:context:target:planet.mars",
            "href": "http://localhost:8080/products/urn:nasa:pds:context:target:planet.
↪mars"
        }
    ],
    "metadata": {
        "label_url": "/data/bundle_insight_rad.xml",
        "update_date_time": "2018-02-01T00:00:00Z",
        "version": "2.1"
    },
    "properties": {
        "pds:Stream_Text.pds:name": [
            "Introduction to the Radiometer Data Bundle"
        ],
        "pds:Modification_Detail.pds:description": [
```


(continues on next page)

(continued from previous page)

```
            "Pre-peer review version",
            "First release",
            "The collections urn:nasa:pds:insight_rad:data_calibrated and␣
↪urn:nasa:pds:insight_rad:data_derived were added to this bundle with InSight Release␣
↪1b.",
            "Changed Observing_System_Component name in this label from RAD to␣
↪RADIOMETER to match context product name. Expanded Citation_Information description."
        ],
        "..."
        "pds:Investigation_Area.pds:type": [
            "Mission"
        ]
    }
}
```

Properties follow the *Fields Dot Notation*.

application/xml

The request:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad::2.1'␣
↪\
   --header 'Accept: application/xml'
```

Returns:

```
<PdsProduct xmlns="http://pds.nasa.gov/api">
    <id>urn:nasa:pds:insight_rad::2.1</id>
    <type>Product_Bundle</type>
    <title>Mars InSight Lander Radiometer Data Archive</title>
    <description/>
    <start_date_time>2018-05-05T00:00:00Z</start_date_time>
    <stop_date_time>3000-01-01T00:00:00.000Z</stop_date_time>
    <investigations>
        <investigations>
            <title/>
            <id>urn:nasa:pds:context:investigation:mission.insight</id>
            <href>http://localhost:8080/products/
↪urn:nasa:pds:context:investigation:mission.insight</href>
            <type/>
            <description/>
        </investigations>
    </investigations>
    <observing_system_components>
        <observing_system_components>
            <title/>
            <id>urn:nasa:pds:context:instrument_host:spacecraft.insight</id>
            <href>http://localhost:8080/products/urn:nasa:pds:context:instrument_
↪host:spacecraft.insight</href>
            <type/>
            <description/>
```

(continues on next page)

(continued from previous page)

```
        </observing_system_components>
        <observing_system_components>
            <title/>
            <id>urn:nasa:pds:context:instrument:radiometer.insight</id>
            <href>http://localhost:8080/products/
↪urn:nasa:pds:context:instrument:radiometer.insight</href>
            <type/>
            <description/>
        </observing_system_components>
    </observing_system_components>
    <targets>
        <targets>
            <title/>
            <id>urn:nasa:pds:context:target:planet.mars</id>
            <href>http://localhost:8080/products/urn:nasa:pds:context:target:planet.mars
↪</href>
            <type/>
            <description/>
        </targets>
    </targets>
    <metadata xmlns="">
        <archive_status xmlns="http://pds.nasa.gov/api"/>
        <creation_date_time xmlns="http://pds.nasa.gov/api"/>
        <label_url xmlns="http://pds.nasa.gov/api">/data/bundle_insight_rad.xml</label_
↪url>
        <update_date_time xmlns="http://pds.nasa.gov/api">2018-02-01T00:00:00Z</update_
↪date_time>
        <version xmlns="http://pds.nasa.gov/api">2.1</version>
    </metadata>
    <properties>
        <pds:Stream_Text.pds:name>Introduction to the Radiometer Data Bundle</pds:Stream_
↪Text.pds:name>
        <pds:Modification_Detail.pds:description>Pre-peer review version</
↪pds:Modification_Detail.pds:description>
        ...
        <pds:Investigation_Area.pds:type>Mission</pds:Investigation_Area.pds:type>
    </properties>
</PdsProduct>
```

Tag names under *properties* follow the *Fields Dot Notation*.

application/vnd.nasa.pds.pds4+json

The request:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad::2.1'␣
↪\
  --header 'Accept: application/vnd.nasa.pds.pds4+json'
```

Returns:

```
{
```

(continues on next page)

(continued from previous page)


```
    "id": "urn:nasa:pds:insight_rad::2.1",
    "meta": {
        "node_name": "PDS_ENG",
        "ops:Label_File_Info": {
            "ops:file_name": "bundle_insight_rad.xml",
            "ops:file_ref": "/data/bundle_insight_rad.xml",
            "ops:creation_date": "2020-01-15T17:40:30Z",
            "ops:file_size": "6805",
            "ops:md5_checksum": "adfd86bbf2573c37d862e27e08f332db"
        },
        "ops:Data_Files": [
            {
                "ops:file_name": "readme.txt",
                "ops:file_ref": "/data/readme.txt",
                "ops:creation_date": "2020-01-03T17:58:09Z",
                "ops:file_size": "1114",
                "ops:md5_checksum": "192de32c12437c180a9e14d60fe4b89a",
                "ops:mime_type": "text/plain"
            }
        ],
        "ops:Tracking_Meta": [
            {
                "ops:archive_status": "archived"
            }
        ]
    },
    "pds4": {
        "Product_Bundle": {
            "Identification_Area": {
                "product_class": "Product_Bundle",
                "Modification_History": {
                    "Modification_Detail": [
                        {
                            "modification_date": "2018-02-01",
                            "description": "Pre-peer review version",
                            "version_id": 0.1
                        },
                        {
                            "modification_date": "2019-04-22",
                            "description": "First release",
                            "version_id": 1
                        },
                        "..."
                    ]
                },
                "information_model_version": "1.11.0.0",
                "logical_identifier": "urn:nasa:pds:insight_rad",
                "version_id": 2.1,
                "Citation_Information": {
                    "publication_year": 2018,
                    "description": "The InSight Radiometer data bundle consists of data␣
↪in three collections:\r\n               data_raw, data_calibrated, and data_derived.\
```


(continues on next page)

(continued from previous page)

```
↪r\n                The bundle also includes the HP3/RAD Software Interface␣
↪Specification in \r\n                the HP3/RAD document collection.",
                    "author_list": "InSight RAD Science Team",
                    "doi": "10.17189/1517568"
                },
                "title": "Mars InSight Lander Radiometer Data Archive"
            },
            "..."
        }
    }
}
```

pds4 property contains a translation in JSON of the PDS4 XML Label.

In addition a *meta* object contains fields related to the managed of the record in the PDS Registry[14], see *ops Namespace* for details.

application/vnd.nasa.pds.pds4+xml

The request:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight_rad::2.1'␣
↪\
  --header 'Accept: application/vnd.nasa.pds.pds4+xml'
```

Returns:

```
<pds_api:product xmlns:pds_api="http://pds.nasa.gov/api" xmlns:ops="https://pds.nasa.gov/
↪pds4/ops/v1">
    <pds_api:id>urn:nasa:pds:insight_rad::2.1</pds_api:id>
    <pds_api:meta>
        <node_name>PDS_ENG</node_name>
        <ops:Label_File_Info>
            <ops:file_name>bundle_insight_rad.xml</ops:file_name>
            <ops:file_ref>/data/bundle_insight_rad.xml</ops:file_ref>
            <ops:creation_date>2020-01-15T17:40:30Z</ops:creation_date>
            <ops:file_size>6805</ops:file_size>
            <ops:md5_checksum>adfd86bbf2573c37d862e27e08f332db</ops:md5_checksum>
        </ops:Label_File_Info>
        <ops:Data_Files>
            <ops:Data_Files>
                <ops:file_name>readme.txt</ops:file_name>
                <ops:file_ref>/data/readme.txt</ops:file_ref>
                <ops:creation_date>2020-01-03T17:58:09Z</ops:creation_date>
                <ops:file_size>1114</ops:file_size>
                <ops:md5_checksum>192de32c12437c180a9e14d60fe4b89a</ops:md5_checksum>
                <ops:mime_type>text/plain</ops:mime_type>
            </ops:Data_Files>
        </ops:Data_Files>
        <ops:Tracking_Meta>
            <ops:Tracking_Meta>
```

(continues on next page)

[14] https://nasa-pds.github.io/registry/

(continued from previous page)

```
                <ops:archive_status>archived</ops:archive_status>
            </ops:Tracking_Meta>
        </ops:Tracking_Meta>
    </pds_api:meta>
    <pds_api:pds4>
        <Product_Bundle
    xmlns="http://pds.nasa.gov/pds4/pds/v1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1 https://pds.nasa.gov/pds4/pds/v1/
↪PDS4_PDS_1B00.xsd">
            <Identification_Area>
                <logical_identifier>urn:nasa:pds:insight_rad</logical_identifier>
                <version_id>2.1</version_id>
                <title>Mars InSight Lander Radiometer Data Archive</title>
                <information_model_version>1.11.0.0</information_model_version>
                <product_class>Product_Bundle</product_class>
                <Citation_Information>
                    <author_list>InSight RAD Science Team</author_list>
                    <publication_year>2018</publication_year>
                    <doi>10.17189/1517568</doi>
                    <description>
                The InSight Radiometer data bundle consists of data in three collections:
                data_raw, data_calibrated, and data_derived.
                The bundle also includes the HP3/RAD Software Interface Specification in
                the HP3/RAD document collection.
            </description>
                </Citation_Information>
               ...
            </Identification_Area>
           ...
        </Product_Bundle>
    </pds_api:pds4>
</pds_api:product>
```

The tag *pds_api:pds4* contains the XML PDS4 label.

In addition a *meta* object contains fields related to the managed of the record in the PDS Registry[15], see *ops Namespace* for details.

application/kvp+json

This format is useful when one only need a few fields from the metadata.

The request:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products?limit=10&fields=lidvid&
↪fields=title' \
  --header 'Accept: application/kvp+json'
```

Returns:

[15] https://nasa-pds.github.io/registry/

```
{
 "summary": {
     "q": "",
     "hits": 17,
     "took": 55,
     "start": 0,
     "limit": 10,
     "sort": [],
     "properties": [
         "lidvid",
         "title"
     ]
 },
 "data": [
     {
         "lidvid": "urn:nasa:pds:insight_rad:data_derived::7.0",
         "title": "InSight RAD Derived Data Collection"
     },
     {
         "lidvid": "urn:nasa:pds:insight_rad:data_raw::8.0",
         "title": "InSight RAD Raw Data Collection"
     },
     "..."
 ]
}
```


Properties follow the *Fields Dot Notation* when they are coming from the PDS4 standard or the *ops Namespace*.

text/csv

This format is useful when one only need a few fields from the metadata.

The request:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products?limit=10&fields=lidvid&
↪fields=title' \
   --header 'Accept: text/csv'
```

Returns:

```
lidvid,title
"urn:nasa:pds:insight_rad:data_derived::7.0","InSight RAD Derived Data Collection"
"urn:nasa:pds:insight_rad:data_raw::8.0","InSight RAD Raw Data Collection"
"urn:nasa:pds:insight_rad:data_derived:hp3_rad_der_00014_20181211_073042::1.0","InSight␣
↪HP3 Radiometer Experiment Derived Product:hp3_rad_der_00014_20181211_073042"
...
```

Open Data

NOT IMPLEMENTED

See <https://project-open-data.cio.gov/>_ and example of application at <https://cmr.earthdata.nasa.gov/search/site/docs/search/api.html#
data>_

2.3.2 ops Namespace

The response content, in addition to the information found in the PDS4 label contains some attributes related to the management of the datasets in the registry.

A dedicated namespace `ops` is used to manage these attributes in the API, for example: *ops:Label_File_Info.ops:file_name* in *Fields Dot Notation* used in JSON or in XML tag *<ops:Label_File_Info><ops:file_name>*.

The list of `ops` attributes is given in the following table:

Class	Attributes	Description	Example
Data_File_Info	creation_date_time		2021-09-10T15:58:03Z
	file_name		collection_document_hp3rad.csv
	file_ref		link[16]
	file_size	in bytes	137
	md5_checksum		cd24cbc46c45ed023f039b3e2beb6606
	mime_type		text/plain
Label_File_Info	creation_date_time		2021-09-10T15:58:03Z
	file_name		collection_document_hp3rad.xml
	file_ref		link[17]
	file_size	in bytes	8655
	md5_checksum		aa584be2cd34d1899f19d39c23cccba1
Harvest_Info	harvest_date_time		2021-11-16T06:03:30.952311900Z
	node_name		PDS_GEO
Tracking_Meta	archive_status		archived

2.3.3 No Results Found

2 cases are considered:

- When there are not results to a **search query, applying parameters to the URL** (e.g. `?q='lid eq fred'`, `keyword`...), you will get an **empty array** (for example *[]* in JSON) as a result.
- When the **URL itself cannot be resolved**, as in `/products/fred` you will get a **404 error** (not found).

2.3.4 Missing Values

Properties with empty or null values should be dropped from the JSON response unless the user asked specifically for the field (through *field* API parameter). In this case the value must be **null**, without quotes.

Rationale

If a property is optional or has an empty or null value, consider dropping the property from the JSON, unless there's a strong semantic reason for its existence (taken from this discussion[18])

Following interactions with OGC/EDR specification group: see ticket[19]

We choose **null** without quotes for missing values of fields explicitly requested by the user.

[16] https://pds-geosciences.wustl.edu/insight/urn-nasa-pds-insight_documents/document_hp3rad/collection_document_hp3rad.csv
[17] https://pds-geosciences.wustl.edu/insight/urn-nasa-pds-insight_documents/document_hp3rad/collection_document_hp3rad.xml
[18] https://softwareengineering.stackexchange.com/questions/285010/null-vs-missing-key-in-rest-api-response
[19] https://github.com/opengeospatial/ogcapi-environmental-data-retrieval/issues/171#issuecomment-767805902

We conform to EDR specification for this aspect, see EDR parameter response[20]

This should not be mistaken for an actual PDS4 value since missing values in PDS4 labels. are detailed with a nil:reason attribute.

2.4 Tutorials/Cookbooks

2.4.1 Sample JupyterLab Notebooks (Python)

The following Git repository contains example JupyterLab notebooks for the application programmer's interface (API) of the Planetary Data System, that can be used as a tutorial to work with the PDS Search API.

https://github.com/NASA-PDS/pds-api-notebook/

2.4.2 Web Search Interface Tutorial (HTML/Javascript)

When developing a web client to the API, if you are not from JPL, contact us (pds_operator@jpl.nasa.gov) so that we can set the CORS attributes for you.

If you are on JPL's network, develop your application locally on port 80 on your laptop connected to the VPN and test your application with URL http://localhost.jpl.nasa.gov/....

An example implementation of an HTML/Javascript client of the API can be found in the github repository https://github.com/NASA-PDS/gis-web-client .

2.4.3 Search Examples

Here are some examples of Search API Recipes:

Search API Cookbook

Recipes for various search scenarios using the PDS Search API.

> **Note**
>
> curl command line tool is used to request the API in this documentation. curl is available in many operating systems by default. If not, you can get curl from https://curl.se/ or using a package management tool specific to your operating system (brew, apt, ...).

Search For Product Versions

Recipes for searching for the latest version of a product, or all versions of a product, including superseded versions.

> **Warning**
>
> Since our servers are not fully populated with all PDS data sets, the examples presented in this user guide may return empty results or 404 (Not Found) errors. If there is a data set you would like added, please submit a request to the PDS Help Desk[21] for assistance.

[20] http://docs.opengeospatial.org/DRAFTS/19-086.html#req_edr_parameters-response
[21] https://pds.nasa.gov/?feedback=true

Search the Latest Version of a Product

To retrieve the **latest** versions of product `urn:nasa:pds:mars2020.spice`, the request is:

The request:

```
https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:mars2020.spice
```

which is equivalent to:

```
https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:mars2020.spice/latest
```

Search for All Versions of a Product

To retrieve **all** the versions of product `urn:nasa:pds:mars2020.spice`, the request is:

```
https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:mars2020.spice/all
```

Search By Specific Metadata

The following recipes describe some example queries of the Search API using the *q*= query parameter showing some more complex use cases for querying PDS data.

> **Warning**
>
> Since our servers are not fully populated with all PDS data sets, the examples presented in this user guide may return empty results or 404 (Not Found) errors. If there is a data set you would like added, please contact the PDS Help Desk[22] for assistance.

Search by Processing Level

Search for the 10 latest collections which processing level is "Raw":

Query: `(pds:Primary_Result_Summary.pds:processing_level eq "Raw")`

Listing 1: curl command

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products' \
    --data-urlencode 'limit=10' \
    --data-urlencode 'q=(pds:Primary_Result_Summary.pds:processing_level eq "Raw")'
```

Search Observational Products by Target

Search for all Observational Products targeting Bennu:

Query: `(ref_lid_target eq "urn:nasa:pds:context:target:asteroid.101955_bennu")`

Listing 2: curl command

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products' \
  --data-urlencode 'q=(ref_lid_target eq "urn:nasa:pds:context:target:asteroid.101955_
↪bennu")'
```

[22] pds-operator@jpl.nasa.gov

Search by Reference

Search all products which are referring to a given LID:

Listing 3: curl command

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products' \
    --data-urlencode 'limit=200' \
    --data-urlencode 'q=((pds:Internal_Reference.pds:lid_reference eq
↪"urn:nasa:pds:context:investigation:mission.orex") or (pds:Internal_Reference.pds:lid_
↪reference like "urn:nasa:pds:context:investigation:mission.orex::*"))' | json_pp
```

Search for DOIs

Digital Object Identifiers[23] are useful to cite the data you are using in your research. DOIs for PDS data are minted for PDS4 Bundles, PDS4 Collections, PDS4 Documents, and PDS3 Data Sets. The level at which the DOI is minted differs from data set to data set.

The following recipes describe how to find a DOI for a particular product or data set in the Search API metadata.

See the [DOI Search](https://pds.nasa.gov/tools/doi/) for an online interface for searching this information.

See the documentation on Citing PDS Data[24] for more information on how to use a DOI to cite your data.

Warning

Since our servers are not fully populated with all PDS data sets, the examples presented in this user guide may return empty results or 404 (Not Found) errors. If there is a data set you would like added, please contact the PDS Help Desk[25] for assistance.

How to Find the DOI associated with an Observational Product

We assume you know the identifier of the product you are working with but a couple are provided in the examples below.

Search for a Product's Collection DOI

Run the following request to get the DOI associated with the collection the observational product *urn:nasa:pds:compil-comet:nuc_properties:description::1.0* belongs to:

Listing 4: curl command

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:vg1-saturn-pos-
↪hgcoords-96sec:data-spice:spice-hg::1.0/member-of' \
    --data-urlencode 'fields=pds:Citation_Information.pds:doi' \
    --header 'Accept: application/kvp+json'
```

You will get the following result:

```
{
    "summary" : {
```

(continues on next page)

[23] https://www.doi.org/
[24] https://pds.nasa.gov/datastandards/citing/
[25] pds-operator@jpl.nasa.gov

(continued from previous page)


```
        "q": "",
        "hits": 1,
        "took": 125,
        "start": 0,
        "limit": 100,
        "sort": [],
        "properties": ["pds:Citation_Information.pds:doi"]
    },
    "data": [
        {
           "pds:Citation_Information.pds:doi":"10.17189/1522962"
        }
    ]
}
```


Note that you might not find any DOI at the collection level, in this case you can try to get the DOI from the bundle.

Search for a Product's Bundle DOI

To get the DOI associated with the bundle the observational product *urn:nasa:pds:insight.spice:document:spiceds::1.0* belongs to:

Listing 5: curl command

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products/urn:nasa:pds:insight.
↪spice:document:spiceds::1.0/member-of/member-of' \
    --data-urlencode 'fields=pds:Citation_Information/pds:doi' \
    --header 'Accept: application/kvp+json'
```

You will get the following result:


```
{
    "summary" : {
        "q": "",
        "hits": 2,
        "took": 135,
        "start": 0,
        "limit": 100,
        "sort": [],
        "properties": [
            "pds:Citation_Information.pds:doi"
        ]
    },
    "data": [
        { },
        {
            "pds:Citation_Information.pds:doi": "10.17189/1517566"
        }
    ]
}
```

How to Find the PDS Product Associated with a DOI

To get the PDS product metadata associated with a the DOI *10.17189/1517568*:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products' \
    --data-urlencode 'q=(pds:Citation_Information.pds:doi eq "10.17189/1522962")' \
    --header 'Accept: application/json'
```

You will get a JSON response of the PDS products (any class of product, for example collections or bundles) which have referenced the given DOI.

You can get the result in different format using content negociation with the Accept header parameter.

Search by Time Range

For example, as a user I need to find the Voyager 1 PWS Spectrum Analyzer CDF files covering March of 1979 so I can make a plot.

Here is one of the XML files, rendered: https://search-pdsppi.igpp.ucla.edu/ditdos/viewFile?id=pds://PPI/voyager1.pws.sa/data/1979/vg1pws_lr_19790105_v5.20.xml .

To build this query, we can search by the instrument and time range:

```
((pds:Time_Coordinates.pds:start_date_time ge "1979-03-01T00:00:00.000Z") and
   (pds:Time_Coordinates.pds:start_date_time lt "1979-04-01T00:00:00.000Z") and
   (ref_lid_instrument eq "urn:nasa:pds:context:instrument:vg1.pws"))
```

Do query that using curl, it would look like this:

```
curl -L --get 'https://pds.nasa.gov/api/search/1/products' \
     --data-urlencode 'q=((pds:Time_Coordinates.pds:start_date_time ge "1979-03-
↪01T00:00:00.000Z") and ' \
     --data-urlencode '(pds:Time_Coordinates.pds:start_date_time lt "1979-04-01T00:00:00.
↪000Z") and ' \
     --data-urlencode '(ref_lid_instrument eq "urn:nasa:pds:context:instrument:vg1.pws"))
↪' \
     --header 'Accept: application/json'
```

Looking for more recipes? Or have some useful recipes of your own? Checkout the PDS API Discussion Board[26] or contact the PDS Help Desk[27]

Looking for more recipes? Or have some useful recipes of your own? Checkout the PDS API Discussion Board[28] or contact the PDS Help Desk[29]

2.5 PDS Search API Links

- Base URL with online documentation: https://pds.nasa.gov/api/search/1/
- User's Manual: https://nasa-pds.github.io/pds-api/guides/search.html
- Python Client: https://nasa-pds.github.io/pds-api-client
- Jupyter Notebook Examples/Tutorials: https://github.com/NASA-PDS/search-api-notebook

[26] https://github.com/NASA-PDS/pds-api/discussions
[27] pds-operator@jpl.nasa.gov
[28] https://github.com/NASA-PDS/pds-api/discussions
[29] pds-operator@jpl.nasa.gov

- YouTube Tutorial: https://www.youtube.com/watch?v=jTclsXR713Y

CHAPTER

THREE

SPECIFICATIONS

Each published version of NASA PDS APIs is documented here:

- Search API v1.5.0
- Search API v1.4.0
- Search API v1.3.0
- Search API v1.1.1
- Search API v1.1.0
- Search API v1.0.0
- DOI API v0.2

More details and rationale for the design can be found in the *general conventions* and the *search API user's guide*.

CHAPTER

FOUR

GENERAL API CONVENTIONS

4.1 Reference Documents

Several websites and documents were used as references for designing this API and the accompanying guidelines, including:

1. Open API Initiative[30]
2. Open APIs Specification[31]
3. Microsoft API Guidelines[32]
4. Microsoft API Design Best Practices[33]
5. NASA Earth Data APIs[34]
6. Google Custom Search REST API[35]
7. EPN-TAP[36]
8. Earth Data Common Metadata Repository (CMR)[37]
9. Swagger for Developing API Spec[38]
10. Open Search[39]
11. Library of Congress Search/Retrieval by URL[40]
12. PDS OPUS API[41]
13. PDS Imaging Atlas API[42]
14. OGC Environmental Data Retrieval[43]

[30] https://www.openapis.org/
[31] http://spec.openapis.org/oas/v3.0.2
[32] https://github.com/Microsoft/api-guidelines/blob/master/Guidelines.md
[33] https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design
[34] https://earthdata.nasa.gov/collaborate/open-data-services-and-software/api
[35] https://developers.google.com/custom-search/v1/using_rest
[36] https://arxiv.org/pdf/1407.5738.pdf
[37] https://cmr.earthdata.nasa.gov/search/site/docs/search/api.html
[38] https://swagger.io/
[39] https://en.wikipedia.org/wiki/OpenSearch
[40] http://www.loc.gov/standards/sru/sru-2-0.html
[41] https://opus.pds-rings.seti.org/apiguide.pdf
[42] https://pds-imaging.jpl.nasa.gov/tools/atlas/api/
[43] https://github.com/opengeospatial/ogcapi-environmental-data-retrieval

4.2 General Applicable Open API Conventions

4.2.1 Specification Standard

The API complies with Open API 3.0.

4.3 Restful Principles

4.3.1 Resources

Resources are coded as URI (e.g. *http://domain/api/pets*). Resources should be nouns (verbs are bad)

4.3.2 Verbs

Users interact with resources through HTTP request verbs[44]. The PDS API uses GET and POST:

- GET is relevant to get resource representation from the API when the extraction criteria is simple.
- POST, in a read-only context, is relevant to provide the API with complex request criteria.

Future iterations of the API will transform it to be an idempotent REST API[45], utilizing GET, PUT, DELETE, HEAD, OPTIONS and TRACE HTTP methods.

4.3.3 Resource Representation

When a HTTP request verb (e.g. GET, POST, etc.) is applied to a resource (e.g. *http://domain/api/pets*) he/she gets a resource representation.

Many flavors of representations are possible to be returned from a single resource. For example: subsets of a whole, formats, versions, etc…

The resource representation should be self-described as much as possible.

They should be wrapped in envelopes which prevent from vulnerabilities linked to the direct access to json arrays in javascript code (see https://haacked.com/archive/2008/11/20/anatomy-of-a-subtle-json-vulnerability.aspx/). A response with this format is fine:

```
{
   "summmary": {"..."},
   "data": ["..."]
}
```

4.4 Other Conventions

Beyond the OpenAPI standard, there are multiple options regarding general design of an API. We primarily use the following source which is very complete and not too dogmatic: https://www.moesif.com/blog/api-guide/api-design-guidelines/

Some peer web API specifications are also considered as references for the design for the PDS API specification:

- ESDIS Common Metadata Repository API[46]
- OGC environment data retrieval[47]

[44] https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them
[45] https://restfulapi.net/idempotent-rest-apis/
[46] https://earthdata.nasa.gov/collaborate/open-data-services-and-software/api/cmr-api
[47] http://docs.opengeospatial.org/DRAFTS/19-086.html

4.4.1 URL Resource Naming: Case

Kebab-case[48] (lower case and hyphens '-' used to fill the spaces in) is used for url resource naming.

For example:

- `discipline-nodes` in http://pds.nasa.gov/api/references/0.1/discipline-nodes

(for the rationale see this stackoverflow discussion[49]

4.4.2 URL Resource Naming: Plural vs Singular

Resources are named plural or singular depending on the use case.

Plural are used when the resources is a collection the user will subset from, for example `/pets/scooby-doo` or `/planets/mars` or '/users/user-id' or 'collections?q=…"

Singular are used when the resource is accessed as one. For example '/profile' to access the profile of the current user.

See this post[50] for more details.

4.4.3 URL Resource Naming: API Versioning

The API will have versions and the deployed versions are likely to be heterogeneous in the PDS system.

Two options have been considered to manage versions (see https://restfulapi.net/versioning/ :

- Version in the URL, e.g. pds.nasa.gov/api/search/1/
- Content negotiation headers (e.g. Accept: application/vnd.example+json;version=1.0)

To keep things as simple as possible, content negotiation will not be used for version management. A server API implementation will implement a single version of the API definition.

However:

- We advise to use the version in the URL of the API when it is deployed, although it is not part of the API definition.
- The version is mandatory in the resource representations (result of a request)

4.5 Pagination/Sort

The query parameters for pagination are:

Parameter	Description
start	Index of first item returned in the response
limit	Maximum number of item expected in the response

See https://www.moesif.com/blog/technical/api-design/REST-API-Design-Filtering-Sorting-and-Pagination/

[48] https://en.wiktionary.org/wiki/kebab_case
[49] https://stackoverflow.com/questions/10302179/hyphen-underscore-or-camelcase-as-word-delimiter-in-uris
[50] https://medium.com/@atomaka/single-and-plural-rails-routes-for-the-same-resource-330d985b6595

CHAPTER

FIVE

TOOLS AND SERVICES

5.1 Search

- Server: https://github.com/NASA-PDS/registry-api
- Client: https://github.com/NASA-PDS/pds-api-client

5.2 DOI

- Server: https://github.com/NASA-PDS/doi-service/
- Client (DOI Editor Web App): https://github.com/NASA-PDS/doi-ui/
- Client (DOI Search Web App): https://github.com/NASA-PDS/wds-react

CHAPTER

SIX

SUPPORT

6.1 Contact Us

Feel free to post a question on the PDS API Discussion Board[51] or submit a request to the PDS Help Desk[52] for any additional questions, comments or concerns.

6.2 Discussions

As part of the PDS API Users community you can use and contribute to the discussion forum[53] hosted on GitHub.

6.3 Contribute

For information on how to contribute to NASA-PDS codebases please take a look at our Contributing Guidelines[54].

6.4 Report a Bug or New Feature Request

- Report a Bug[55]
- Report a New Feature[56]

[51] https://github.com/NASA-PDS/pds-api/discussions
[52] https://pds.nasa.gov/?feedback=true
[53] https://github.com/NASA-PDS/pds-api/discussions
[54] https://github.com/NASA-PDS/.github/blob/main/CONTRIBUTING.md
[55] https://github.com/NASA-PDS/pds-api/issues/new?template=bug_report.md
[56] https://github.com/NASA-PDS/pds-api/issues/new?template=feature_request.md

CHAPTER

SEVEN

ACKNOWLEDGEMENTS

The PDS Search API is designed by a working group involving all the nodes of the Planetary Data System.

7.1 PDS API Working Group

- Mcclanahan, Timothy (PDS PO)
- Lynn Neakrase (ATM)
- Ed Guiness (GEO)
- Tom Stein (GEO)
- Dan Scholes (GEO)
- Mark Bentley (ESA)
- Myche McAuley (IMG)
- In Sook Moon (PPI)
- Rob French (RMS)
- Matt Tiscareno (RMS)
- Conor Kingston (SBN)
- David Chang (SBN)
- Daniel Darg (SBN)
- Emily Law (EN)
- Vivian Tang (EN)
- Al Niessner (EN)
- Ramesh Maddegoda (EN)
- Thomas Loubrieu (EN)
- Jordan Padams (EN)
- Boris Semenov (NAIF)