

Data Access meeting 2017-01-30

Date

30 Jan 2017

Attendees

- [Gregory Dubois-Felsmann](#)
- [Fritz Mueller](#)
- [John Gates](#)
- [Kenny Lo](#)
- [Brian Van Klaveren](#)
- [Trey Roby](#)
- [Tatiana Goldina](#)
- [Unknown User \(ymei\)](#)
- [Kian-Tat Lim](#)
- (please fill in any I've missed!)


Goals

- SLAC-IPAC meeting to refine the DAX APIs - this time directed at near term requirements for PDAC


Discussion items

Time	Item	Who	Notes
------	------	-----	-------

metaserv for PDACv1	Brian Van Klaveren	<ul style="list-style-type: none"> • Brian is bringing <code>metaserv</code> back into operation for PDACv1. This is involving recovering the code that was used in the 2015 Bremerton end-to-end-test era, and adapting it to the current configuration of databases and services. • At last week's Data Access meeting and PDAC meeting, we began to discuss the API that already existed, the need to confirm what it will be for the current PDACv1 deployment, and to discuss some issues that Brian identified with its previous state. We planned to discuss these today. • The discussion centered around the API page on Confluence. • Brian mentioned three major issues today: <ul style="list-style-type: none"> ◦ The existing <code>metaserv</code> as-implemented API is missing all the image metadata query endpoints (M9-M13). ◦ There seems to be some confusion about the roles of the "level" and "database" levels of the URI pathname hierarchy in the API. ◦ He would like to introduce new host-based endpoints to allow direct query of specific servers. • Re: image metadata queries - Gregory noted that in PDACv1 SUIT is currently using explicit <code>dbserv</code> queries against the known image metadata tables in order to find images, so for the moment we have no urgent need for this functionality. In the longer run it is probably reasonable to have a <code>dbserv</code>-independent way of querying the catalog of images - after all, it is logically coherent for there to be an image service in the absence of a database service. (Not mentioned in this meeting, but in discussions earlier last week: there is also no assurance that a <code>dbserv</code> query could return the specific URI for an image found in an image metadata table, or even guarantee that an image referred to in the table actually is available for service.) Gregory noted that M12 and M13 are different from all the other originally proposed M* API endpoints, because they return information about individual items of data that are in LSST's holdings, as opposed to information about the <i>structure</i> of the holdings (database schemas, collection names, etc.). This might be somewhat counterintuitive as a result, and we could consider whether to consolidate image-existence queries with image-retrieval queries in <code>imgserv</code>. But this needs deeper thought. We agreed to table this question for now and to return to a wider discussion of image catalog queries and wildcarding, and APIs that can return multiple images (by reference or by value), at a later date. • "level" and "database" in the <code>metaserv</code> URI pathnames: <ul style="list-style-type: none"> ◦ There are several related concerns here: <ul style="list-style-type: none"> ▪ The presence of the <code>dc</code> level in the list on the API page raised questions, because one might think that, most of the time, a "data challenge" would be a (perhaps partial) dress rehearsal of some specific part of the DM processing, i.e., Level 1 (nightly) or Level 2 (DRP), and that therefore a data challenge might be modeled as an "unofficial release" at the corresponding level. ▪ The pathname component following "level" is called "database" which implies that it is an actual database name, but the sample values for level "L2" are given as "DR1" and "DR2". This begs the question of whether all the tables corresponding to a given data release are all required to be in the same database. ▪ It also begs the question of whether the simple user-friendly names like "DR1" will be actual database names or, in some way, aliases to the actual names. This touches on issues of how a data release will actually be built and released and whether that process can guarantee the database name to be something so simple and user-friendly. Another way of phrasing the question is whether the values returned from API M4 are guaranteed to be directly usable in SQL /ADQL queries in <code>dbserv</code>. It seems likely that they should be, if any form of generic table handling is to be available in the SUIT (as must be needed for at least the EFD service). ◦ We decided that most of these issues need not be resolved immediately for PDACv1, but should be given attention soon from an architectural perspective. • For immediate purposes, we decided that the PDACv1 and PDACv2 SDSS Stripe82 and WISE/NEOWISE data should be modeled as level=L2, as the Object and ForcedSource tables are characteristic of Level 2. It was proposed that the data-release identifier (the next, "database", pathname component) be something simple and friendly like "PDACv1", but this may not be realistic given the above discussion of whether this pathname component is exactly equal to an actual database name. This component may also be needed for distinguishing between the SDSS and WISE data in PDAC. We asked the DAX group to come back with a specific proposal for how to use the "database" (i.e., data-release) component for PDAC. • We did not discuss the host-based endpoints further during the meeting. They do not appear to be a PDACv1 need from the SUIT side. • We then went on to discuss specifics of the M1-M8 API for database metadata. <ul style="list-style-type: none"> ◦ SUIT is not likely to use M1-M7 <i>in the current version of PDAC</i>, though they will be needed later. As a result, we did not discuss them in detail. <ul style="list-style-type: none"> ▪ (added by Gregory Dubois-Felsmann during editing:) We might use M7 to retrieve descriptions of the current tables, if substantive user-facing descriptions are available. ▪ Kian-Tat Lim asked about the implication (in the non-use of M6) that SUIT is hard-coding table names. SUIT admitted to this practice, for PDACv1, but noted that <i>something</i> must eventually be hard-coded somewhere to encode the <i>semantics</i> of tables. It is not possible to provide a scientifically informative UI without knowing, for instance, the meaning of the foreign-key relationship between the Object and ForcedSource tables. If someone has selected an Object, we will want to offer them a "show light curve" UI action, and this cannot be implemented without encoding somewhere the knowledge that light curves for an Object live in the ForcedSource (in PDACv1) and Source (to be added later) tables. A later meeting will have to address a design for how to encode this metadata without requiring the SUIT to depend directly on the table names involved. ◦ Functionality related to M8 is of immediate interest to SUIT for PDACv1. <ul style="list-style-type: none"> ▪ The use of the current M8 itself would require parsing SQL, so of even more immediate interest would be endpoints (which we might call M8a, M8b, etc.) that return: <ul style="list-style-type: none"> • Column descriptions (eventually it would be nice if this made available both a short description and a link to more extensive documentation) • UCDs • Units ▪ (added by Gregory Dubois-Felsmann during editing:) Pre-parsed information on available foreign-key relationships. (not needed for PDACv1) ▪ We discussed whether an additional API level that allowed the addressing of metadata queries to single columns would be useful, but the immediate SUIT consensus was that we would prefer just to query on entire tables. ▪ Tatiana Goldina raised the question of how we will get information on the structure of the result tables from dynamic operations like joins. Gregory Dubois-Felsmann noted that this would have to include, for instance, UCDs and units for the return values of <code>qserv_</code> and <code>scisql_</code> functions in order to really cover all the use cases. We agreed that this is not a problem we can hope to solve for PDACv1 and deferred this to a later date. • Kenny Lo requested that the next version of the API definition include a "BNF-like" definition of the APIs, distinguishing between required syntax and keywords and syntactic variables, providing a more formal definition of the API in addition to the existing examples.
---------------------------	--------------------------	---

<imgserv </imgserv enhancem ents for PDACv1, mainly cutouts from specified exposures	John Gates, Kian- Tat Lim	<ul style="list-style-type: none"> The SUIT folks summarized the immediate need for an API endpoint for single-epoch image cutouts, to support the light-curve viewer. This application allows a user to view a photometry table, a time-series plot of the photometry (either in absolute time, e.g., MJD, or phase-folded), and postage-stamp images around the object for every point in the light curve. The ideal behavior would be: <ul style="list-style-type: none"> Allow the specification of a particular image ("calexp" or, more formally for LSST, "Processed Visit Image" (PVI)) by ID, or by broken-down ID (like the current interim <code>ids</code> interface) Allow the center of the cutout to be expressed in sky coordinates Perform the cutout without re-projection (e.g., scaling or rotation) of any kind, so that the actual pixel values of the single-epoch image are returned to SUIT, and the sides of the cutout rectangle are aligned with the pixel grid <ul style="list-style-type: none"> Note that the inclusion of a WCS with the cutout then allows SUIT to rotate images to a common coordinate system if this is requested by the user in the UI. Allow the lengths of the sides of the cutout rectangle to be expressed in angular units on the sky (NB: this is not the same as specifying the box corners in sky coordinates) We deferred to a later date discussion of whether in the long run this needs to be an asynchronous interface. Kian-Tat Lim suggested that, rather than continue with the interim <code>id</code> and <code>ids</code> APIs that were provided for the first version of PDAC <code>imgserv</code>, we return to the originally proposed API style of I6-I10. After discussion, we agreed to this plan even for the near-term implementation, provided that the "interim" API endpoints are preserved as well for a period of overlap, to ease the complexity of deployment. We discussed the "<code>?plane={data,mask,variance}</code>" behavior of the proposed APIs. SUIT stated that this capability is not currently needed and that it would be acceptable for <code>imgserv</code> to simply return the natural three-plane results of the cutout operation by default. The API page should be changed to indicate that this is the default when the "<code>?plane=</code>" is not supplied. (There was some confusion over the originally intended difference between I6 and I7. It may be that they can be edited to show that I6 is explicitly a single-plane request, and that I7 is a multiple-plane request, which can either omit the qualifier altogether, or take a "<code>?plane=data,mask</code>" type of qualifier.) <ul style="list-style-type: none"> tl;dr: SUIT would like I6 without "<code>?plane=</code>". For cutouts, the qualifier syntax is neither precisely that of I8 or that of I10. SUIT proposes that both of the following be accepted, though the first is the most urgently needed: <ul style="list-style-type: none"> <code>GET /image/v0/L2/DR1/calexp/12345/cutout?ra=1&dec=1&widthAng=10&heightAng=10</code> (where the <i>ra</i> and <i>dec</i> are in decimal degrees and the <i>heightAng</i> and <i>widthAng</i> are in arc seconds) <code>GET /image/v0/L2/DR1/calexp/12345/cutout?ra=1&dec=1&widthPix=30&heightPix=30</code> (where the <i>ra</i> and <i>dec</i> are in decimal degrees and the <i>heightPix</i> and <i>widthPix</i> are in pixels) The height and width are the full dimensions of the sides of the cutout, not the distance from the center to the sides. No "<code>?plane=</code>" is required for the time being - three-plane cutouts work for SUIT for PDACv1. The current behavior of the cutout service for dealing with regions of the cutout rectangle that do not exist on the image is acceptable (return an image of the full requested size, with mask bits set to identify the blank/cropped region). SUIT asks that DAX clearly document the behavior when the requested cutout center is not in the specified image (details of this question expanded in editing by Gregory Dubois-Felsmann): is this an error, or will the system return a cutout as long as there is even one pixel of overlap (i.e., at a corner) between the image and the cutout rectangle? If there is no overlap at all, is an error returned or just an all-blank, all-masked image? If it is an error, is it distinct from the error returned if the specified image does not exist? (added by Gregory Dubois-Felsmann during editing:) Note that for "calexp" images we generally will refer to them by ID, whether performing a cutout or not. For "coadd" images the most common (but not only) use case eventually will be to retrieve them purely by cutout coordinates, without an ID, given that the natural concept of the LSST coadds is that of quasi-all-sky images. The <code>imgserv</code> APIs should clearly handle both cases. This means a different endpoint, without the ID in the path specification but, presumably, with "/cutout" as its last component, is required. This does not have to be solved immediately for PDACv1. As above, we will follow Kenny Lo's recommendation that the API specification include a more formal grammar in addition to the examples. SUIT also needs a mechanism for requesting a cutout from a specific image identified by broken-down ID. We were not able to get to the discussion of a proposed form for this, even whether it is a single API call or not. Note that a broken-down ID request carries with it the potential for missing components of the ID, requiring the interface to treat that as either an error or an implicit wildcard. There is a JIRA ticket for the work to provide a near-term cutout service: <div data-bbox="448 1232 1227 1402" style="border: 1px solid orange; padding: 10px; margin-top: 10px;">  DM-9115 - Jira project doesn't exist or you don't have permission to view it. </div>
Planning subseque nt meetings		<ul style="list-style-type: none"> We are looking forward to getting proposals for the immediate needs of SUIT by Wednesday of this week: <ul style="list-style-type: none"> Resolution of what the "level" and "database" elements of the <code>metaserv</code> pathnames will be Additional M8-level API endpoints for column descriptions, UCDS, and units The specified-image cutout APIs SUIT will read/think about them and we will meet again on 06 Feb 2017 to decide how to proceed. We will use a meeting after that - 13 Feb 2017 or 27 Feb 2017 (2/20 is a holiday) to begin work on the numerous tabled issues for the future, assuming that the near-term development is proceeding smoothly.

- In addition to

 **DM-9115** - Jira project doesn't exist or you don't have permission to view it.

, note that there are also several

existing epics and tickets relating to the refinement of the DAX APIs.

Action items

- ☐ [Brian Van Klaveren](#) 02 Feb 2017 Provide a draft of the near-term resolution for the use of the "database" field in the `metaserv` pathnames in PDACv1 (the proposal should handle both the SDSS and WISE data), assuming a common "level=L2" for all PDACv1 data.
- ☐ [Brian Van Klaveren](#) 02 Feb 2017 Provide a draft of the additional M8-like endpoints for descriptions, UCDs, and units.
- ☐ [John Gates](#), [Kian-Tat Lim](#) 02 Feb 2017 Review the proposed single-ID cutout APIs above and comment. Update API page with the result.
- ☒ [Kian-Tat Lim](#), [John Gates](#) 06 Feb 2017 Make a proposal for how to support cutouts from `calexps` identified by broken-down ID. Added: Kenny Lo

is working this under a to-be-expanded



DM-9929 - Jira project doesn't exist or you don't have permission to view it.