Development Notes: Restores of calibrated ALMA MOUSes for the AAT/PPI

What is a Restore, and why is it 'A Thing(tm)'?

After an observation (or a set of observations, in the case of ALMA) are complete, the data is then calibrated using the CASA Pipeline. Unfortunately, due to space constraints, the entire Calibrated Measurement Set (or CMS for short) is far to large to store in the archive due to the details of how CASA lays out and uses the data. Instead, NRAO does the next best thing: We archive what are called the 'Calibration Tables', these are far smaller sets of data which describe the corrections applied to the data by CASA to achieve the CMS.

The application of an already calculated calibration solution is significantly faster (by a factor of ~10 or so) than performing a full re-calibration of the data. In addition, the restored CMS is either identical (or nearly so) to the initial result. This provides a measure of reproduce-ability which is important for the purposes of the Science Ready Data Products initiative.

The reason that a CMS (and thus a restore) are important is that it is the starting point for data analysis. What this provides is freedom from some grunt-work which underlies the real scientific analysis. Most critically, any imaging or statistical analysis of the data starts with a CMS. Once the restore process is stable and validated, it can be used as the precursor to much more scientifically interesting analysis.

Why were restores ALMA developed so much later than those for the EVLA?

Unlike the process for the EVLA, performing a restore for ALMA is slightly more complicated. In many cases, ALMA processing involves more than one Execution Block (EB) at a time, and thus all the original EBs must be acquired along with the calibration products. In addition, the PPR for handling ALMA data needs to meet the requirements of a much more stringent schema than is used for EVLA data processing.

The typical 'work unit' for ALMA is the Member Observation Unit Set (MOUS). This is an organizational structure for data (see: here) which is used for automated calibration and imaging using the CASA Pipeline, and each MOUS has a unique identifier to by which the results of those pipeline runs can be found.

For data processed after October 2017, it is far more reasonable to perform an MOUS level restore because the calibration products are stored separately from the imaging products within the ALMA NGAS system, thus greatly simplifying the organizational work involved. There are plans for products ingested earlier to be split up and reingested, but we have been give no timescale for when that project might start.

When can the AAT/PPI perform a restore?

The AAT/PPI performs periodic checks for new ALMA observations in the NAASC metadata database, and performs its own metadata ingestion to provide access to ALMA observations via the system's tools. This process has recently been expanded to include checking for new calibrations which have been completed. Metadata pertaining to calibrations which are appropriate for the AAT/PPI automated restore process is then ingested and made available via the search interface.

A calibration in the NAASC database is considered 'complete' when there are files of class 'science' ingested (typically, these are images resulting from the calibrated data). However, not all ALMA calibrations can be handled via the CASA pipline. In some instance (15-20%) some human intervention is required in order to produce an appropriate calibration for the data. These calibrations include special files which indicate that work beyond the pipeline was required. Because the AAT/PPI does not have the same level of DA support as the NAASC, those calibrations are excluded from our restore system. The link in the paragraph above leads to a more detailed discussion of the ALMA reingestion process used by the system.

How is an ALMA restore requested?

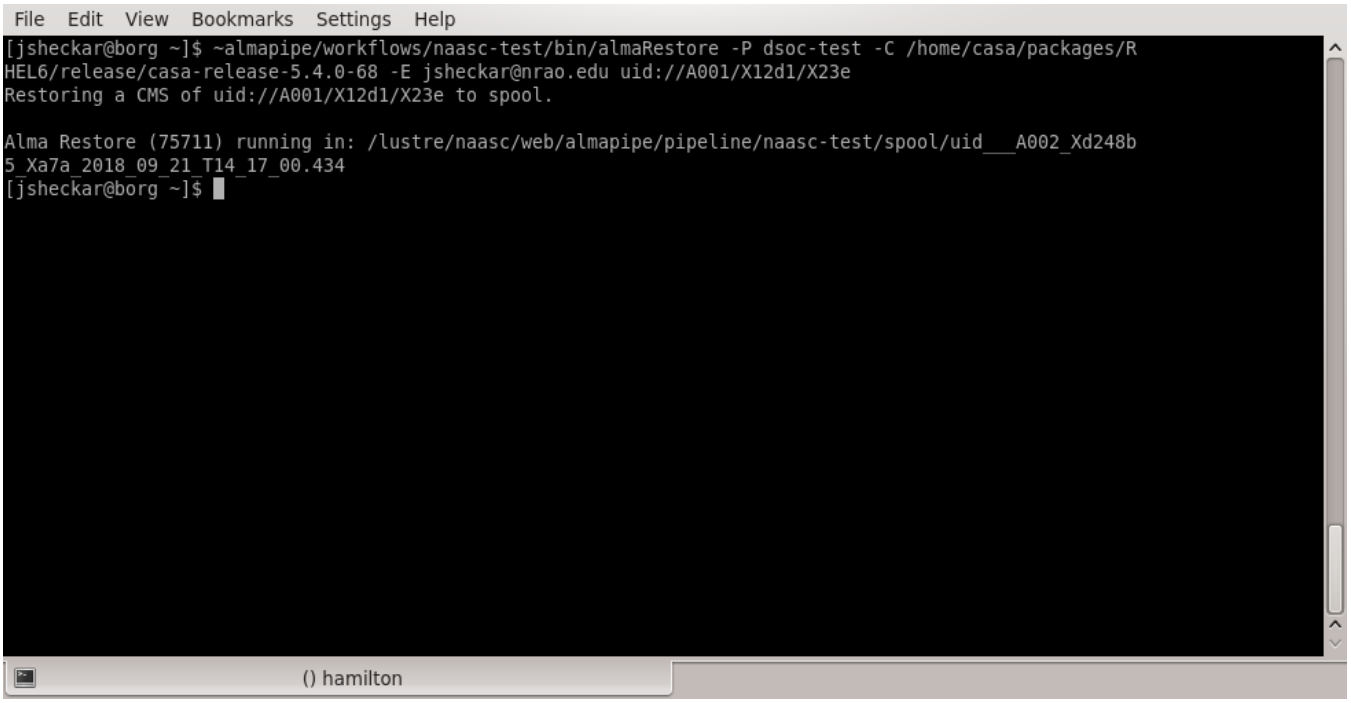
Fundamentally, a restore requires the MOUS UID (for example: uid://A001/X1284/X265f, uid://A001/X12a3/X80e, uid://A001/X12cc/X4a, or uid://A001/X1284/X266), with which the other relevant data can be extracted from the ALMA metadata database. In the context of the AAT/PPI, there are some technical issues which end up necessitating some other information.

The almaRestore Command:

Used for initial testing, this initiates a restore of the requested MOUS via our workflow system. The tool looks up a couple of extra pieces of information (the project code, and an ASDM which belongs to the MOUS) in the background and then sends the following event to initiate the restore workflow:


```
{
    "eventName": "runAlmaBasicRestoreWorkflow",
    "type": "edu.nrao.archive.workflow.messaging.commands.StartWorkflow",
    "additionalPaths": [],
    "metadata": {
        "workflowName": "AlmaOusRestoreWorkflow",
        "processingSite": "NAASC",
        "deliveryFormat": "CMS",
```

```
        "telescope": "ALMA",
        "fileSetIds": ["uid://A002/Xd248b5/Xa7a"],
        "ousStatusId": "uid://A001/X12d1/X23e",
        "projectCodeOrDataType": "2017.1.00370.S",
        "cliCorrId": "f1a4dd4a-73ae-4aa7-ac30-9397d31dadab",
        "casaHome": "/home/casa/packages/RHEL6/release/casa-release-5.4.0-68"
    }
}
```

This particular restore requested a particular CASA version (casaHome's value) to be used, since there is a policy disconnect between EVLA and ALMA about what constitutes an officially acceptable CASA version. The cliCorrId is generated and used internally to return information to the command runner about where their data processing is happening. If an email is specified, the an email is sent to that address when the workflow completes.

The other fields in the StartWorkflow are standard values to allow the workflow to proceed properly.

NOTE: The command was run in Charlottesville, but it specified the equivalent CAPO profile for Socorro systems. That is due to a limitation imposed by the messaging system which underlies the AAT/PPI functionality. The workflow tracking will not happen properly unless the message is sent to the main messaging hub in Socorro.

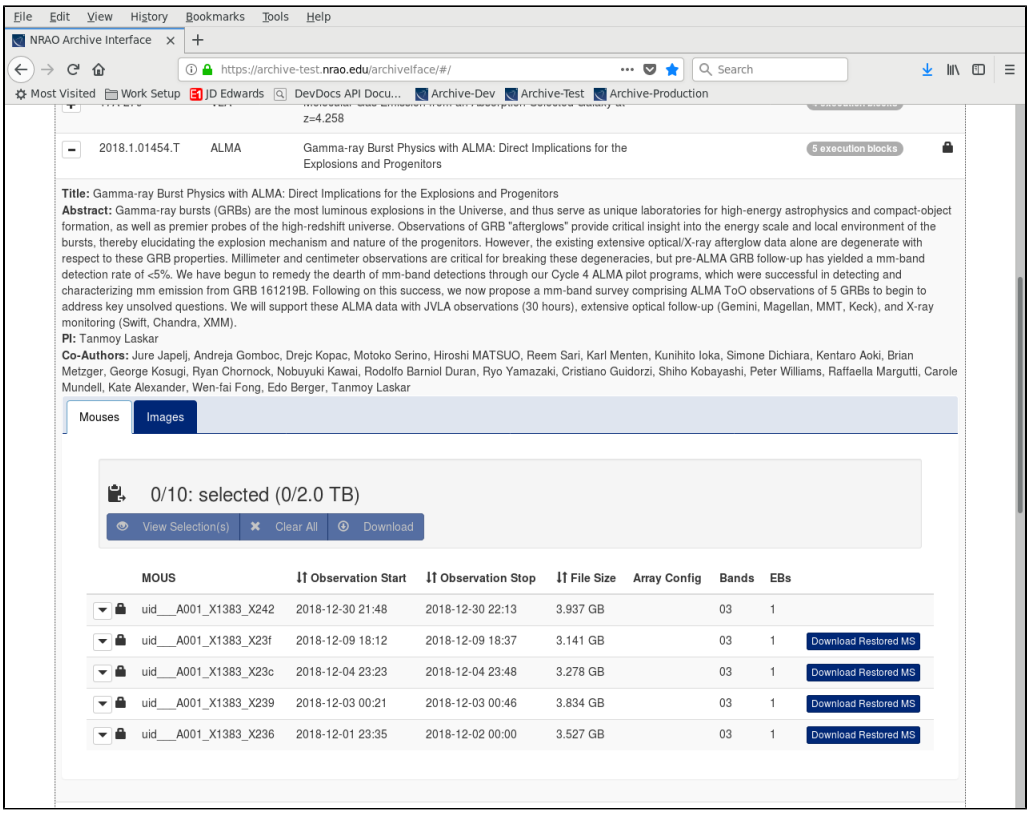
What if I only have an EB (or ASDM) UID?

You can retrieve the MOUS UID from a given ASDM UID via the following query:

```
select sbs.MOUS_STATUS_UID FROM BMMV_SCHEDBLOCK sbs join ALMA.SHIFTLOG_ENTRIES shifts ON shifts.SE_SB_ID = sbs.
ARCHIVE_UID join AQUA_EXECBLOCK ebs ON ebs.EXECBLOCKUID = shifts.SE_EB_UID where ebs.EXECBLOCKUID='....';
```

The AAT/PPI Front End:

The search interface for the AAT/PPI has been updated to handle ALMA MOUS structures, and indicates whether a restore can be requested with the existence of a button underneath the 'cals' column. This specialized button results in a workflow start event much like the one above, but with additional fields in the metadata pertaining to the final delivery location of the calibrated MSes and a few additional fields used by other workflows initiated via the search interface. It is important to note that even if ALMA data has been calibrated and imaged, that does not mean that the data are appropriate for an automated restore process. Please see the section above for how we determine which MOUSes are restore candidates.



The archive front end (via the request handler) sends a similar (although larger) set json command for the workflow initiation. It includes a set of parameters to specify how to delivery the results of the workflow, and also some extraneous information which is mostly of interest for other workflows.

What does the AAT/PPI ALMA restore workflow do?

In rough terms, an ALMA restore is largely similar to an EVLA restore:

1. Set up directory structure & metadata.json
2. Retrieve rawdata
3. Retrieve & stage calibration products & manifest
4. Write the restore PPR
5. Run CASA
6. Deliver the Calibrated Measurement Set

Several of these steps are going to differ in the details, however. The details of obtaining data from the NAASC NGAS machines and the inputs to CASA are variations of the procedure used for the EVLA. In contrast, preparing an appropriate PPR is a greater challenge in the ALMA processing case.

Retrieve raw data from the NAASC NGAS machines:

Only those execution blocks which have a QA0 status of Pass are used in the ALMA processing system, and thus it is only those execution blocks which are required to perform a restore. From the MOUS UID we are given to restore, we can obtain a list of ASDM UIDs via:

```
SELECT EXECBLOCKUID FROM ALMA.AQUA_EXECBLOCK ebs
JOIN ALMA.SHIFTLOG_ENTRIES shifts ON ebs.EXECBLOCKUID = shifts.SE_EB_UID
JOIN ALMA.BMMV_SCHEDBLOCK  sbs    ON shifts.SE_SB_ID = sbs.ARCHIVE_UID
WHERE ebs.QA0STATUS = 'Pass'
AND sbs.MOUS_STATUS_UID ='.....';
```

That list are then fed sequentially into the asdmExportLight script, which is part of the Alma Common Software suite. See the alma-datafetcher.sh script for the appropriate configuration steps. Place the ASDMs underneath the rawdata subdirectory of our working directory as normal.

Retrieve calibration products & manifest from the NAASC NGAS machines:

For the handling of ALMA calibrations, there is the fetchAlmaCals tool. This worker script interrogates the ASA_PRODUCT_FILES table for the files relevant to a restore of that MOUS (those of the 'calibration' and 'script' FILE_CLASS in particular) and performs a verified extraction of these files from the NAASC NGAS system into the products subdirectory of our working directory.

Once the files are extracted from NGAS, there are a pair of additional steps which are done in preparation:

1. decompress and expand the *.hifa_calimage.auxproducts.tgz file in order to provide access to the files it contains. The pipeline will not automatically handle these files being contained in a tar archive (the other .tgz files can be left alone)
2. the *.pipeline_manifest.xml file must be copied over to the rawdata directory, as that is where CASA expects it to be located.

There is a potential alternative solution: it should be possible to make use of the 'exportProducts' tool which is part of the Alma Common Software suite to achieve similar results, but that method was unsuccessful during the prototyping phase and was abandoned for the time being.

NOTE: The master NGAS server name & port number have been extracted out into capo (the almaNgasSystem properties). Those data were taken from the configuration file provided by Rachel Rosen. Updated values can be obtained from /home/acs/config/archiveConfig.properties (accessible from most, if not all, CV machines), in case something changes and our properties get out of date.

Write the restore PPR:

For ALMA data, the PPR must meet the requirements of a stricter schema (defined in the Alma Common Software) than is used for EVLA processing. In particular, the ProjectStructure section is required, and there are different formats for the DataSet section which must be followed. In addition, there is a substructre to the MOUS (called sessions) which must be properly laid out in the ProcessingIntents section.

Empty PPR structure

```
<?xml version="1.0" encoding="UTF-8"?>
<SciPipeRequest xmlns:ent="Alma/CommonEntity"
    xmlns:val="Alma/ValueTypes" xmlns:prp="Alma/ObsPrep/ObsProposal"
    xmlns:orv="Alma/ObsPrep/ObsReview"
    xmlns:ps="Alma/ObsPrep/ProjectStatus"
    xmlns:oat="Alma/ObsPrep/ObsAttachment"
    xmlns:prj="Alma/ObsPrep/ObsProject"
    xmlns:sbl="Alma/ObsPrep/SchedBlock"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="SciPipeRequest">
    <SciPipeRequestEntity entityId="UID_UNASSIGNED"
        entityTypeName="SciPipeRequest" datamodelVersion="0.1"/>
    <ProjectSummary>
        <ProposalCode></ProposalCode>
        <ProposalTitle></ProposalTitle>
        <Observatory></Observatory>
        <Telescope></Telescope>
        <ProcessingSite></ProcessingSite>
        <Operator></Operator>
        <Mode>CSV</Mode>
        <Version>Undefined</Version>
        <CreationTime></CreationTime>
    </ProjectSummary>
    <ProjectStructure>
       <ObsUnitSetRef entityId=""
            partId="" entityTypeName="ObsProject"/>
        <ObsUnitSetTitle>Undefined</ObsUnitSetTitle>
        <ObsUnitSetType>Member</ObsUnitSetType>
        <ProjectStatusRef entityId=""
            entityTypeName="ProjectStatus" documentVersion="1"/>
        <OUSStatusRef entityId="" entityTypeName="OUSStatus"/>
    </ProjectStructure>
    <ProcessingRequests>
      <ProcessingRequest>
    <RootDirectory></RootDirectory>
    <ProcessingIntents>
          <Intents>
            <Keyword>PROCESS</Keyword>
            <Value>true</Value>
          </Intents>
          <Intents>
            <Keyword>SESSION_1</Keyword>
            <Value></Value>
          </Intents>
          <Intents>
            <Keyword>SESSION_2</Keyword>
            <Value></Value>
          </Intents>
          <Intents>
            <Keyword>SESSION_3</Keyword>
            <Value></Value>
```

```
          </Intents>
          <Intents>
            <Keyword>SESSION_4</Keyword>
            <Value></Value>
          </Intents>
          <Intents>
            <Keyword>INTERFEROMETRY_STANDARD_OBSERVING_MODE</Keyword>
            <Value>Undefined</Value>
          </Intents>
        </ProcessingIntents>
    <ProcessingProcedure>
      <ProcedureTitle>hifa_restore_aat_ppi</ProcedureTitle>
          <ProcessingCommand>
            <Command>hifa_restoredata</Command>
        <ParameterSet/>
          </ProcessingCommand>
    </ProcessingProcedure>
    <DataSet>
          <SchedBlockSet>
            <SchedBlockIdentifier>
              <RelativePath></RelativePath>
              <SchedBlockRef entityId=""
                             entityTypeName="SchedBlock" documentVersion="1"/>
              <SBStatusRef entityId="" entityTypeName="SBStatus"/>
              <SBTitle>Undefined</SBTitle>
              <AsdmIdentifier>
                <AsdmRef>
                  <ExecBlockId></ExecBlockId>
                </AsdmRef>
                <AsdmDiskName></AsdmDiskName>
              </AsdmIdentifier>
              <AsdmIdentifier>
                <AsdmRef>
                  <ExecBlockId></ExecBlockId>
                </AsdmRef>
                <AsdmDiskName></AsdmDiskName>
              </AsdmIdentifier>
              <AsdmIdentifier>
                <AsdmRef>
                  <ExecBlockId></ExecBlockId>
                </AsdmRef>
                <AsdmDiskName></AsdmDiskName>
              </AsdmIdentifier>
              <AsdmIdentifier>
                <AsdmRef>
                  <ExecBlockId></ExecBlockId>
                </AsdmRef>
                <AsdmDiskName></AsdmDiskName>
              </AsdmIdentifier>
            </SchedBlockIdentifier>
          </SchedBlockSet>
        </DataSet>
      </ProcessingRequest>
    </ProcessingRequests>
    <ResultsProcessing>
        <ArchiveResults>false</ArchiveResults>
        <CleanUpDisk>false</CleanUpDisk>
        <UpdateProjectLifeCycle>false</UpdateProjectLifeCycle>
        <NotifyOperatorWhenDone>false</NotifyOperatorWhenDone>
        <PipelineOperatorAdress>Unknown</PipelineOperatorAdress>
    </ResultsProcessing>
</SciPipeRequest>
```

Above is the basic layout of the restore PPR. Much of the data simply needs to be added to the correct area (RootDirectory, ProjectCode, etc).

Queries for basic PPR data:

- ProjectSummary

        - ProposalCode
            - available via the ProjectData object
            - select distinct PROJECT_CODE from ASA_PROJECT join ASA_SCIENCE on ASA_PROJECT.PROJECT_UID = ASA_SCIENCE.PROJECT_UID where MEMBER_OUSS_ID='.....';
        - The rest can be handled with default values or 'unknown'
    - ProjectStructure
        - ObsUnitSetRef – This refers to the Project + the partId corresponding to this MOUS within that project.
            - entityId
                - select OBSUNITSETUID from AQUA_OUS where OUS_STATUS_ENTITY_ID = '.....';
            - partId
                - select OBSUNITSETPARTID from AQUA_OUS where OUS_STATUS_ENTITY_ID = '.....';
        - ProjectStatusRef
            - entityId
                - select OBS_PROJECT_STATUS_ID from OBS_UNIT_SET_STATUS where STATUS_ENTITY_ID='.....';
        - OUSStatusRef – This is just our MOUS of interest
            - entityId
                - .....
    - ProcessingRequests
        - RootDirectory
            - Path to the processing area (i.e. spool directory). **NOTE:** This is typically overriden by the SCIPIPE_ROOTDIR environment variable.
        - ProcessingProcedure
            - For a restore, this is typically just a call to the hifa_restoredata pipeline procedure, with no other tasks to perform.
            - However, in the future we will need to handle parameters to the hifa_restoredata call at some point (see the gainmap issue for VLASS

SSA-4893 - Jira project doesn't exist or you don't have permission to view it.

). Some of these types of details are handled in the restore script, but parsing that is beyond the scope of this project.
    - DataSet
        - RelativePath
            - Path from the RootDirectory above to the directory housing products/rawdata/working
        - SchedBlockRef
            - entityId
                - select distinct SCHEDBLOCK_UID from ALMA.ASA_SCIENCE where MEMBER_OUSS_ID = '.....';
                - There is the possibility of multiple SBs linked to an MOUS. The DOMAIN_ENTITY_STATE column of the SCHED_BLOCK_STATUS table should be able to tell us which we want.
        - SBStatusRef
            - entityId
                - select STATUS_ENTITY_ID from SCHED_BLOCK_STATUS where DOMAIN_ENTITY_ ID='SchedBlockRef entityId';
        - AsdmIdentifer
            - Fill out one of these blocks with the ASDM_UID & the santized version for each EB in the MOUS.

ProcessingIntents for the PPR:

In an EVLA PPR, the ProcessingIntents section is simply a portion of XML provided by the pipeline team (or an external user) and placed into the appropriate location in the PPR. For ALMA data, however, there is more information required. ALMA configurations are not nearly as static as those for the EVLA, such that there might be antenna moves which occur between two execution blocks for the same MOUS being observed. So each MOUS can be broken down into one or more 'sessions' of observing. Each session has a label, and it contains 0 or more execution blocks. Since the calibration tables are collected into .tgz files according to the session to which they pertain, getting the correct mapping of session name to ASDM(s) is important.

Because the session numbering scheme takes into account observations with QA0 statuses of 'Failed' and 'SemiPass', a fuller account of the ASDMs which make up the MOUS to be restored needs to be considered. The details of the algorithm are discussed here.

Beyond a Basic Restore:

After a large scale refactor of the PPR handling capability of the AAT/PPI it will be possible to insert extra steps (splits of the resultant MS, etc) for a more customized restored result. For now, the focus of the system is on the delivery of a calibration Measurement Set.

Once the PPR is complete and written out to the working subdirectory, the workflow is ready to proceed.

Run CASA:

There are a few differences in the details of running the CASA pipeline for an ALMA restore. Firstly, we need to use the 'runpipeline.py' script instead of the 'runvlapipeline.py' which is our normal method of invoking casa. This is due to differences in how those two scripts interpret the PPR, but I don't know the details. As a consequence of the first change, we need to define 3 environment variables:

a. SCIPIPE_ROOTDIR This should point to the overall working area (i.e. the spool directory)
b. SCIPIPE_LOGDIR This should point to a 'Logs' directory beneath SCIPIPE_ROOTDIR
c. SCIPIPE_SCRIPTDIR This points to the pipeline recipes directory for the version of CASA+Pipeline we are using

Once those are set, we can invoke CASA with (to be complete):

> *${CASA_HOME}/bin/casa --nogui --nologger --pipeline -c ${CASA_HOME}/pipeline/pipeline/runpipeline.py PPR.xml*

This will be wrapped in an xvfb-run command as normal in a workflow. These changes have been encapsulated in the casa-alma-pipeline.sh script.

Deliver the Results:

In the almaRestore version of the process, there is nothing further done. The calibrated MS is left as it was generated in the working subdirectory, and (if requested) an email is sent to the provided email address to announce completion.

For the external request for an ALMA restore, the system follows the patterns created by the download and custom calibration workflows:

- Check for indications that the CASA pipeline encountered an error
- Move the Measurement Sets into the products subdirectory
- Deliver the products subdirectory to the specified location (typically a download location), at which point the UI is updated to show a link to the results.