***DEVELOP - This is the current "DEVELOP" version of this specification, please use comments or "suggesting" mode only.***

# Green Box Use Cases

These use cases are functionality needed to have a functioning green box (analysis environment). Here we make choices on how processes in the green box will work given our current understanding of the needs of the community.

Associated files of importance:

- HCA Storage System Index, Query, and Eventing Functional Spec & Use Cases: this includes a writeup of what an example Green Box query might look like given the sample data bundles we currently have.
- HCA Storage System Data Bundle Use Cases and Project SOP: includes a little more background on queries.
- Blue Box meeting notes for 20170616 includes discussions around the queries Green box might want to do from Blue Box.
- https://github.com/HumanCellAtlas/data-bundle-examples: our repo for sample bundles.

## Detailed Use Cases

### **Use Case 100**: Run Analysis Workflow and Upload Results

**Priority: Must Have (A)**
**Primary Actor**: "green box" system for analysis, where results will be validated by a "purple box" and pushed to "blue box" storage
**Brief:**
"Green box" engine listens for appropriate message, adds message to launcher queue, selects and performs workflow, and submits output data bundle. The green box will submit derived files from analysis, derived qc products, the analysis.json, logging for the creation of a new bundle once in the Blue Box. Before being stored in the blue box the Purple box verifies/validates the data, accessions the data with external archive IDs if appropriate, validate files as well as the schema of the analysis.json file. Details on the hand-off to purple are detailed in green box use case 109: Clean-up storage of outputs.

If it is not possible to distinguish a reanalysis request from an analysis request the green infrastructure will need a way to find out if any temporary environment change occurred for the pipeline (Use Case 107: Temporary Pipeline Change) and use those changes in the case of a reanalysis.
**Query Type:** Stored query in blue.

**Query:** On the creation or update of an ingest bundle we would need an event to be created based on the sample.donor.species and the assay.single_cell.method.

## **Use Case 101:** Updating a Pipeline Version

**Priority:** Must Have (A)
**Primary Actor:** “green box” engineers, optionally Analysis Working Group, optionally the DCP steering committee, querying into blue to get all the bundles needed to re-run.
**Brief:**
Given semantic versioning of pipelines (MAJOR.MINOR.PATCH), how will we handle updating pipelines.
**Query Type:** See use case 102
**Query:** See use case 102
Future points to clarify:

- Will these updates that change science be coordinated after a release?
    - If the update happens between releases, we will need to reanalyse all data immediately.
- Will the data versions from previous release(s) be re-run with this new pipeline?

### **Sub-Use Case:** Patch update:

A backwards compatible bug fix that does not affect science.

- Will initially (minimally) need to run a regression suite, and gold standard benchmarking tests to confirm science does not change and benchmarks only improve.
- Onboarding optional. As onboarding is automated it may be convenient to put through a onboarding service.
- Portability required. When a portability service exists, the workflow will be ran on gold standard data to check engineering changes did not affect portability.
- No update for AWG. We will not need to update the analysis working group given no science is changed.
- No analysis rerun. No need to rerun previous analysis given science should not have changed.

### **Sub-Use Case:** Minor Update:

A backwards compatible functionality involving code around around the science of the pipeline that should not change the output science.

- Will initially (minimally) need to run a regression suite, and gold standard benchmarking tests to confirm output science does not change and benchmarks only improve.
- Onboarding optional. As onboarding is automated it may be convenient to put through a onboarding service.
- Portability required. When a portability service exists, the workflow will be ran on gold standard data to check engineering changes did not affect portability.

- Update for AWG required. We will need to update the analysis working group given code around science is changed.
- No analysis rerun. No need to rerun previous analysis given output science should not have changed.

### **Sub-Use Case:** Major Update:

Not-backwards compatible changes, either functionality or scientifically. This would be characterized as an update to the pipeline that is large enough to change the results of the gold standard tests (even for the better).

- Onboarding required. Full onboarding will be required.
- Portability required. When a portability service exists, the workflow will be ran on gold standard data to check engineering changes did not affect portability.
- AWG approval required. We will need to update the analysis working group and request approval given science is changed. We will work with the Analysis Working Group to validate the update to gold standard tests and the associated evaluation tool.
- Analysis considered. The request to rerun all relevant data in the DCP will be requested given science should have changed and the data will be inconsistent between workflow versions. The request will be sent to the DCP steering committee. The DCP Steering group will need to confirm we can rerun all affected samples. See green use case 102: Reanalysis of a Pipeline/Data Type.

## **Use Case 102:** Reanalysis of a Pipeline/Data Type

**Priority: Must Have (A)**
**Primary Actor:** green team for formulating queries for reanalysis if needed, DCP steering to approve mass reanalysis, “green box” system for analysis (where results will be validated by a “purple box” and pushed to “blue box” storage).
**Brief:**
This use case describes the scenario where an analysis has already occurred on a file but analysis needs to occur again, potentially due to an update to a data file. This can happen due to a metadata change, a data change (like a top-off event or update of a data file), a reference update, or a pipeline update.

In the below descriptions the term “reanalysis” is used. This is defined exactly the same as an analysis event described in use case 100: Run Analysis Workflow and Upload Results. The reason this can work is that analysis always generates a new analysis bundle which is linked to the version of the ingest bundle that triggered the analysis. In reference to an ingest bundle, the latest analysis bundle that references it produced by a specific computational method and reference will be considered the latest analysis.

When a reanalysis is required the following will occur:

- The green box team will query the blue box to determine how many ingest bundles will be reanalyzed.
- The green box team will notify the DCP steering and request permission.
- The green team will query the blue box for events to trigger reanalysis.

**Query Type:** Manual to blue.
**Query:**

- How many ingest bundles satisfy a query based on sample.donor.species and assay.single_cell.method.
- Create events for all ingest bundles that satisfy a query based on sample.donor.species and assay.single_cell.method.

### Sub-Use Case: Reanalysis due to Metadata update

On first version of the system if any metadata is updated, the event to green triggers reanalysis. As an optimization and depending on the metadata update, it will be explored how to copy forward the metadata changes with the previous analysis bundle analysis outputs. When this type of “copy-forward” analysis occurs, it will be important to indicate it in the analysis.json. This optimization relies on the ability for the green box to know what was updated in the metadata.

### Sub-Use Case: Reanalysis due to a data update

A data file update should always trigger a reanalysis event; in the blue box this must always trigger an event. An example of this use case is a top-off event.

### Sub-Use Case: Reanalysis due to Pipeline Version Update

Not all changes to a pipeline require reanalysis (see use case 101: Updating a Pipeline Version). If an update were to happen and tests showed a change to the scientific outputs (given gold standard tests), reanalysis must occur to make sure the data in the DCP is not inconsistent.

## Use Case 103: Terminate On-going Analysis

**Priority: Must have (A)**
**Primary Actor:** “green box” admin/OP
**Brief:**
This describes the scenario when an analysis does not fail on its own but needs to be terminated. As an example: if we have an abnormally long running pipeline which needs to be investigated.

- An abnormality is identified in the dashboard.
- The green box OPs team has admin power to kill analysis through the dashboard.
- The green box OPs team will review submission.

- A submission will either be re-submitted to the launch service or sent to data brokers depending on the issue. If sent to a data broker the analysis will occur in the future based off of an event from blue after the ingest bundle is updated.
- Zendesk (ticketing system) can be used to communicate with users (internal and external).

**Query Type:** None
**Query:** None

## Use Case 104: Rejection of Malformed Analysis Request

**Priority: Must have (A)**
**Primary Actor:** “green box”, green OPs, data broker, potentially green PO
**Brief:**
This describes the scenario when an issue is found with the analysis request or bundle. This scenario could be due to information in the notification not being accurate, or files pointed to not being available. Bundle issues could be due to missing files that are recorded to exist but do not or incomplete bundles where the bundle contains all files as expected but more files should be available (singleton files that should have a paired file). This will most likely be identified in the launcher or during the workflow.

- The abnormality is identified in the dashboard (via failures monitor).
- The green box OPs team will review submission.
- A submission will be sent to data brokers depending on the issue.
- Zendesk (ticketing system) can be used to communicate with users, data submitters (internal and external).
- The green box OPs team evaluates issue and considers if this event could be caught in the future programmatically and if so submits to green team PO.
- Analysis will occur in the future based off of an event from blue after the ingest bundle is updated.

**Query Type:** None
**Query:** None

## Use Case 105: Recover Analysis Run Failure

**Priority: Must have (A)**
**Primary Actor:** Green OPs, potentially brokers, potentially green PO
**Brief:**
This describes the scenario when an analysis is ran but fails. Here, we would want to troubleshoot the cause for failure and, if possible, re-run the analysis. If the analysis cannot be recovered, it would be sent back to data brokers or the ingest team as appropriate. If the failure occurred within the green box, the failure would be recorded in the green box metrics (i.e. workflow failure rate over time) and this information could be displayed in the HCA dashboard. This would not result in blacklisting of a sample but could result in an update and new analysis event.

- The failure is caught by the failures monitor and dashboard is updated.

- The green box OPs team is notified/uses dashboard to initiate investigation.
- If the investigation concludes with a pipeline issue the green team is involved to update process following use case 106: Park Data While Troubleshooting.
- If the investigation concludes with a problem with the data bundle the data bundle is passed off to a broker. Analysis will occur in the future based off of an event from blue after the ingest bundle is updated.
- The green box OPs team evaluates issue and considers if this event could be caught in the future programmatically and if so submits to green team PO.

**Query Type:** None
**Query:** None

## Use Case 106: Data Will be Queued While Troubleshooting

**Priority: Should have (A)**
**Primary Actor:** Green OPs, potentially green PO, potentially green Comp Bios, potentially AWG
**Brief:**
This describes the scenario where an analysis has failed due to a required pipeline update, or due to services that green box depends on have an outage. This use case assumes that the data received by the green box is ok and does not need to be sent back to a broker. The failed workflow will be handled like all failed workflows (Use Case 105: Recover Analysis Run Failure), and the green box operations team will send a notification back into that workflow queue. The queue will be on hold until the pipeline is restarted.

- The pipeline is updated as defined in use case 102: Reanalysis of a Pipeline/Data Type.

**Query Type:** None
**Query:** None

## Use Case 107: Temporary Pipeline Change

**Priority: Must Have (A)**
**Primary Actor:** Green OPs, possible green PO sign-off
**Brief:**
In this scenario, after an analysis failed and was recovered, there is the possibility of a need for a temporary change to the pipeline to accommodate that piece of data. The likely reason for this use case is a case where a specific sample data needs more memory to get through the pipeline. Whatever change is needed would not effect the science being done, and is not a change we want to make to the blessed pipeline.

- In general, environment variables will be parameterized as inputs into the pipelines.
- All inputs to pipelines will be recorded with other metadata in the green box database that tracks state of green box jobs and will be contained in the analysis.json. These changes to an environment variable can be recovered in either location.

- Analysis runs will check environmental parameters for updates before running. (use case 100: Run Analysis Workflow and Upload Results)

**Query Type:** None
**Query:** None

## Use Case 108: Run Test on Pipeline

**Priority: Must Have (A)**
**Primary Actor:**
**Brief:**
We intend to have an automated testing environment to allow for thorough testing. Different types of testing will include the following categories:

- On boarding tests include:
    - Running pipeline tests to make sure they currently run as expected.
    - Running unit tests for tools in the pipeline made as part of the pipeline (as opposed to being an external tool that is called).
    - Benchmarking pipeline on gold standard.
    - Performance of the pipeline is measured including cost, runtime, and memory usage.
- Nightly tests of each pipeline. This will be using a very small, toy data set or abridged gold standard data set. This test will purely focus on making sure each run makes the same output consistently each night. Performance will be measured here to also make sure no there are no changes in performance characteristics but this will not be the main performance test.
- Portability. As a long-term goal to have a service to check if pipelines can run in multiple environments. As this service is designed and implemented, pipelines will be required to adhere to the standards created for portability.
- Performance reporting. As a long-term goal reports will be created that display on boarding characteristics of pipelines.The analysis.json has performance metrics, one can potentially create report using these metrics showing data size vs time per tool. This can be displayed on a reporting dashboard.
    - It would eventually be interesting to draw from multiple data set sizes and plot each step  by input data size to get a data driven measurements of the scaling characteristics of each pipeline task.

**Query Type:** None
**Query:** None

## Use Case 109: Clean-up storage of outputs

**Priority: Must Have (A)**
**Primary Actor:** green and purple
**Brief:**

Green will be it's own broker and will submit to ingestion API. *First iteration: Hand URLs to same API a user would use to send to staging for now

- Green outputs to scratch space
- Post for analysis bundle submission envelope to ingest.
- Ingest gives envelope ID.
- Ingest posts to staging to get staging tmp URL.
- Ingest gives temp URL.
- We igress (intracloud copy) to staging tmp URL.
- Ingestion validates and submit to DSS.
- After submission to DSS we can clean up our scratch space.

The eventual goal is to be an activity that is the staging for this purple interaction will be in place and not require transferring files. No matter the location of staging, files will be submitted to purple validation processes, this will return if those files are valid. On passing files are pushed to blue box as a list of cloud paths to a new data bundle to the storage system blue box.
**Query Type:** None
**Query:** None

## Use Case 110: Create Resource Bundle

**Priority: Must Have (A)**
**Primary Actor:**
**Brief:**
Resource bundles will need to be created for the pipelines per reference species and version. Although, initially analysis will focus on homo sapiens, other species or versions may need to be created. It is important we have a pipeline that generates these bundles as needed.

- A bundle will be created with the appropriate reference information.
- On ingest it will be indicated as a resource bundle for an assay.
- An event will fire and trigger the green box.
- The resource bundle will be created with a pipeline as an analysis bundle.
- This would have json files including analysis.json.

**Query Type:** None
**Query:** None

## Use Case 111: Change Reference for Analysis Pipeline

**Priority: Must Have (A)**
**Primary Actor:** "green box"
**Brief:**
This use case describes the scenario where the reference for an analysis pipeline needs to be updated. It is not preferable to perform this update between releases. This should occur immediately after a release occurs. Given the real time nature of the analysis pipelines, this may

not be possible and some samples may be processed with an older reference and need to be updated to a new analysis.

- Resource bundle is created beforehand (see Use Case: Create Resource Bundle).
- Launcher config file is updated for the new reference or reference version.
- Analysis.json reflects the resource bundle used.
- Is there a need to check to reanalyze files already ran or will there be a rule this only occurs immediately after release and, if that is true, do we rerun all other samples in the blue box already ran in previous releases. (question swings back to re-analysis use case above)

**Query Type:** None
**Query:** None

## Use Case 112: Quick QC Preanalysis

**Priority: Could Have (C)**
**Primary Actor:** green box, purple and brokers
**Brief:**
As the green box, we want to run a quick quality control pipeline on a sample before running the entire HCA pipeline on the sample. This idea is modeled after RapidQC used by the Broad Institute to screen genomes for bad quality prior to spending the money to fully analyze them. This will be a very rare event that is informed from prior runs of the pipeline. We will always attempt to run data, even “bad” and place it in the blue box for others to analyze. This is more of the case that there is some artifact in the data that prevents analysis.

This requires

- A quick QC pipeline for each full pipeline to quickly analyze the data in a sufficiently scientifically accurate way
- Ability to decide if a sample should not be ran from not passing QC
    - Could send back to submitter and double check they want to run it.
    - If it fails QC, send back to user facing portal that the sample failed?
- If it does pass QC, what is the process for storing that information and launching the full analysis? Does that intermediate quick QC information need to be stored if it passes? Can this decision be made within green box and full analysis be launched automatically after passing?

**Query Type:** None
**Query:** None

## **Use Case 113:** Run Batch Analysis Workflow and Upload Result

**Priority: Could Have (C)**
**Primary Actor:** green box and release group
**Brief:**

There are “green box” workflows that may need to be run for cohort-level analysis. The analysis will be performed on a group of data that is generated and uploaded separately, i.e. are not all in the same bundle. The idea is that a metadata/release manager would occasionally update a cohort manifest. Upon version update to this file type/file UUID an event is triggered that a listener in the green box can subscribe to. The update essentially triggers the re-run of a workflow on the files in the manifest, and this updates the results for a particular cohort. It assumes a human is involved in updates and, therefore, ultimately is the one triggering these cohort/study-level analysis events. These can be coupled to releases, for example.

**Pre-requisites:**

- Individual bundles of data files are uploaded/updated and assigned UUIDs

**Requirements:**

- User submits a file that provides metadata about the UUIDs of data files/bundles to be processed, perhaps updating an existing manifest in an existing data bundle or creating a new data bundle.
- A component for the green box is watching for events for this file type/data bundle and triggers the appropriate workflow based on the content of the updated manifest file.
- Results from the workflow are uploaded back to the blue box, to a new data bundle or possibly and update to an existing data bundle.

**Query Type:** Manual (performed by the green infrastructure).
**Query:** Select all ingest bundles associated with a project.

## Use Case 114: Storing State in the Green Box

**Priority: Should have (B)**
**Primary Actor:** green box, operations, user-facing portal
**Brief:**
As the green box operations team or a user of the HCA DCP, I would like to know where in the DCP my data is. Inside of green box, I would like to know when my data is received by green, queued, or running in a workflow, or when it is submitted back to ingest. This requires green box to store state information about where data is within green. It also requires the information to be available for a user-facing portal and an operations dashboard to access.
Level of resolution of state: Waiting in {queue_name}, Launching to {pipeline}, Running on {pipeline}, Submitting to ingest, complete, Error, Error (investigating).
**Query Type:** Manual (available to others infrastructure).
**Query:** There would be queries about state that could happen in any box.

- What is the state of {input_bundle}?

## **Use Case 115** - Packaged Imaging Parameters

**Priority: Should Have (C)**
**Primary Actor: Green Box and Purple Box**
**Brief:**

Analysis of imaging data can be highly parameter-dependent, and setting these parameters often requires the subjective judgment of an expert. For example, a standard image based transcriptomics pipeline depends on image pre-processing, which in turn depends on several threshold and filter parameters. In practice, an appropriate parameterization is often determined by a scientist through visual inspection. Small changes to the parameters can significantly alter the outcome of an analysis. Furthermore, the parameters may vary across labs and assays due to variations in hardware and chemistry. The Green Box should be able to detect a file of suggested parameters in a submitted bundle, use these parameters when running the analysis, and record the parameters in analysis.json.
**Query Type:** Made by Green Box when notification is received
**Query:** Is there a parameters.json in the bundle?

## **Use Case 116** - Multi'omics Analysis

**Priority: Should Have (B)**
**Primary Actor: Green Box**
**Brief:**
The same biological specimen can be analyzed using different experimental methods. For example, a single specimen may be divided in two and submitted for scRNA-seq and image-based transcriptomics. These two experiments will likely be done separately and at different times, but a secondary analysis method could make use of both data types simultaneously. Since the sequencing and imaging data will be prepared by different people at different times, they will show up in the Blue Box as separate bundles. The Green Box should be able to detect if there is data in the Blue Box that would allow for a joint analysis workflow to be run.

**Query Type**: Made by the Green Box when notification is received.
**Query**:

1. Is there a data bundle in the Blue Box where specimen_id={same id} and assay_method={complementary method}.
2. Does that data bundle have an analysis bundle where the computational_method={complementary type + version}?