Sequel® System SMRT® Link Web Services API Use Cases v5.1.0

Pacific Biosciences

Chapter 1: Introduction 3

Chapter 2: Use Cases 4

1.0 How to Obtain Reports for a Specific Job 4
1.1 Workflow 4
1.2 Diagram 4
1.3 Example 5
2.0 How to Obtain Reports for a Specific Dataset 5
2.1 Workflow 5
2.2 Diagram 6
2.3 Example 6
3.0 How to Obtain QC Reports for a Specific Run 6
3.1 Workflow 6
3.2 Diagram 8
3.3 Example 8
4.0 How to Obtain QC Reports for a Specific Collection 9
4.1 Workflow 9
4.2 Diagram 10
4.3 Example 10
5.0 How to Obtain Recent Runs 10
5.1 Workflow 10
5.2 Diagram 11
5.3 Example 11
6.0 How to Set Up a Run Design 12
6.1 Workflow 12
6.2 Diagram 12
6.3 Example 13
7.0 How to Monitor Run Progress 13
7.1 Workflow 13
7.2 Diagram 14
7.3 Example 14
8.0 How to Capture Run-Level Summary Metrics 16
9.0 How to Set Up a Job on a Specific Collection 16
9.1 Workflow 16
9.2 Diagram 17
9.3 Example 18
10.0 How to Delete a Job 21
10.1 Workflow 21
10.2 Diagram 22
10.3 Example 22
11.0 How to Set Up an Analysis Job for a Specific Pipeline 23
11.1 Workflow 23
11.2 Diagram 24
11.3 Example 25

Chapter 1: Introduction

The SMRT Link Web Services API, provided by Pacific Biosciences, allows integration of SMRT Link with third party software, as well as accessing features such as designing and performing QC on instrument runs, querying new data from the instrument, and starting analyses on the sequence data.

This document describes common tasks performed using the SMRT Link Web Services and provides "how to" recipes for accomplishing these tasks. To accomplish a task, you usually need to perform several API calls; the workflow describes the order of these calls.

The Web Services support RESTful access from web clients, and can be used from the command-line with `wget` or `curl`, from scripting languages such as PHP, Python, PERL, and from programming languages such as Java, C++, and C#.

The API includes functions for:

- Managing run designs;
- Managing resources, such as instruments;
- Managing Data Sets;
- Managing Projects;
- Managing analysis jobs;
- Monitoring the health status of the system.

The Web Services APIs:

- Run in or under a standard Linux/Apache environment, and can be accessed from Windows, macOS or Linux operating systems.
- Are installed as part of the SMRT Analysis, and require a one-time configuration.

For detailed information on the SMRT Link Web Services API calls, see

`http://<SMRTLinkServername:Portnumber>/sl/docs/services/`

where `<SMRTLinkServername:Portnumber>` is the name and port number of your local SMRT Link server.

Note: This document uses `9091` as the services port number in all the examples, as this is the default for the smrtlink installer.

Chapter 2: Use Cases

1.0 How to Obtain Reports for a Specific Job

1.1 Workflow

To obtain the reports for a job, given the job ID, perform the following steps:

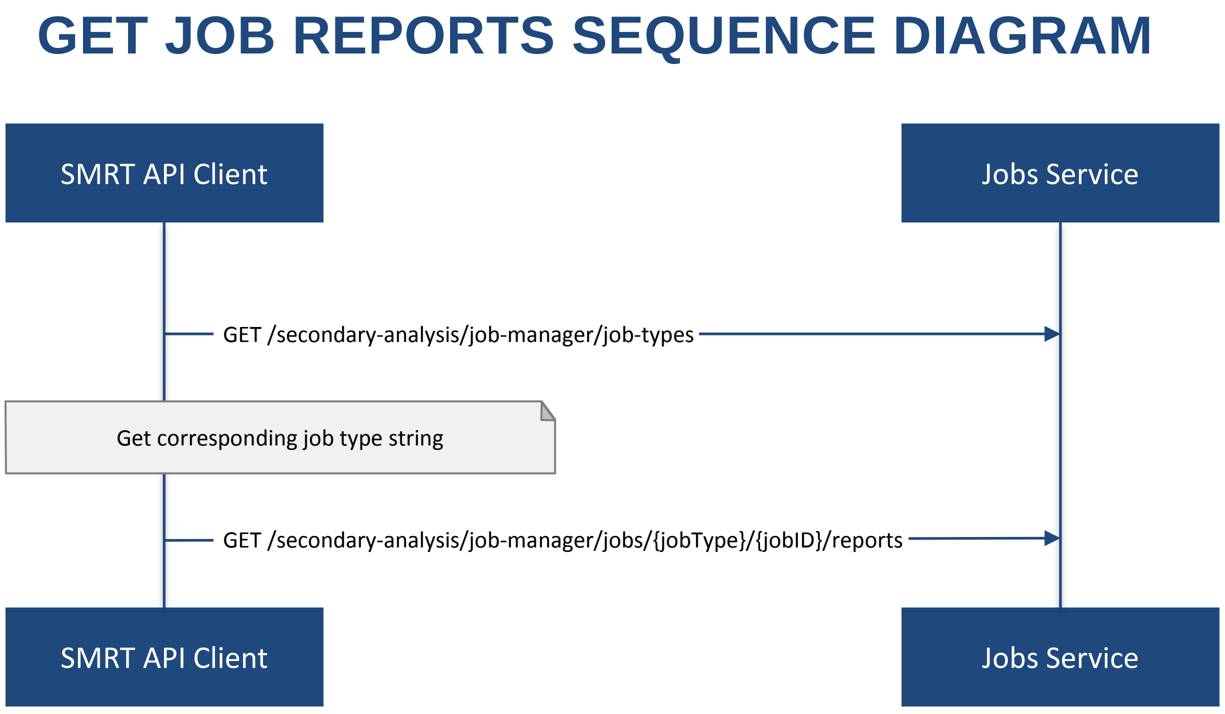
1. Determine the job type from the list of available job types. Use the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/job-manager/job-types
```

2. Get the corresponding job type string. (The job type can be found in the `jobTypeId` field.)
3. Get reports produced by the job.
   Given the job ID and the job type, use them in the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/job-
manager/jobs/{jobType}/{jobID}/reports
```

1.2 Diagram


GET JOB REPORTS SEQUENCE DIAGRAM
SMRT API Client
Jobs Service
GET /secondary-analysis/job-manager/job-types
Get corresponding job type string
GET /secondary-analysis/job-manager/jobs/{jobType}/{jobID}/reports
SMRT API Client
Jobs Service

1.3 Example

Suppose you view a SMRT Analysis job results page in the SMRT Link UI.
To find the job ID, look for the “Analysis ID” field under Analysis Overview, Status.

Note: The job ID will also appear in the `{jobID}` path parameter of the SMRT Link UI URL

```
http://smrtlink-release:9091/#/analysis/job/{jobID}
```

Suppose you view the following SMRT Analysis job results page:

```
http://SMRTLinkServername.domain:9091/#/analysis/job/3957
```

Then the job ID is 3957.

To get the job type, use the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/job-manager/job-types
```

Look for the appropriate jobTypeId in the response.

A SMRT Analysis job corresponds to the ‘pbsmrtpipe’ type, so the jobTypeId will be “psmrtpipe”. The desired endpoint is:

```
http://SMRTLinkServername.domain:9091/secondary-analysis/job-
manager/jobs/pbsmrtpipe/3957/reports
```

Use the GET request with this endpoint to get reports produced by the job with ID = 3957:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/job-
manager/jobs/pbsmrtpipe/3957/report
```

2.0 How to Obtain Reports for a Specific Dataset

2.1 Workflow

To obtain reports for a dataset, given the dataset UUID, perform the following steps:

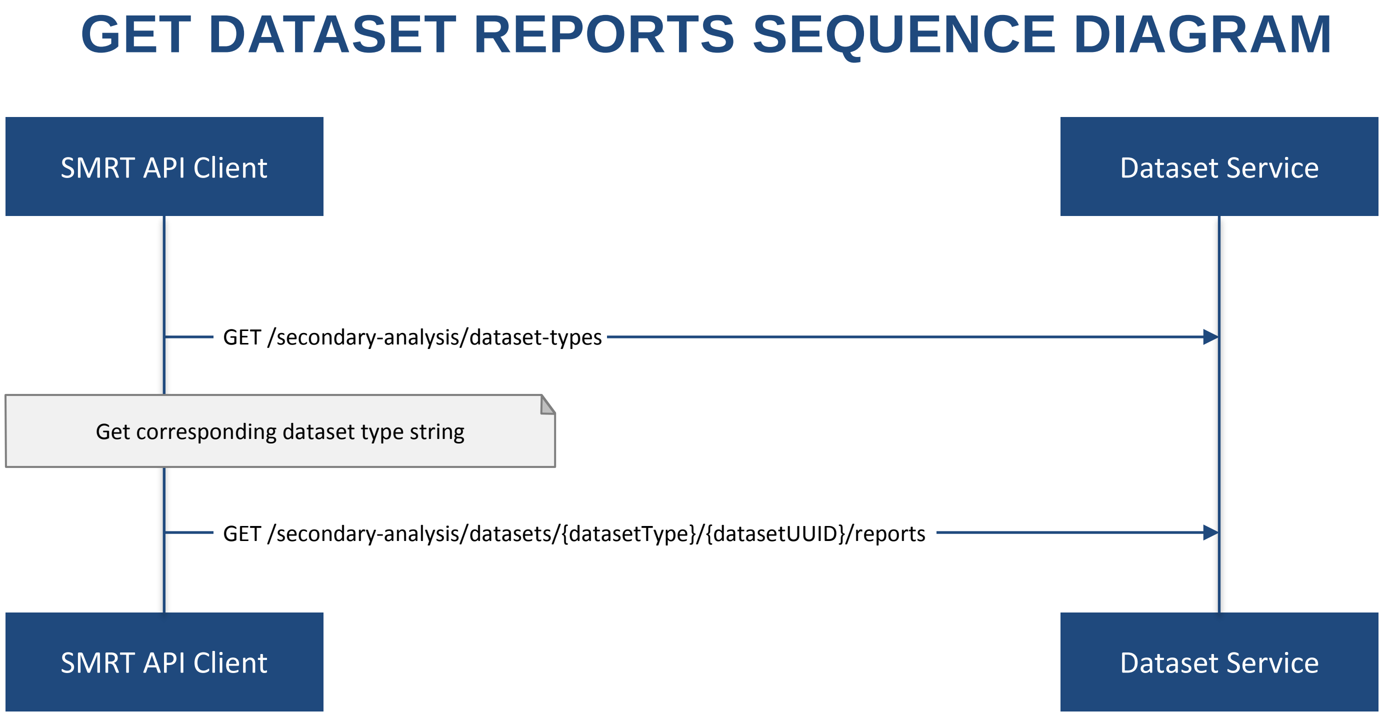
1. Determine the dataset type from the list of available dataset types. Use the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/dataset-types
```

2. Get the corresponding dataset type string.
    - The dataset type can be found in the "shortName" field.
3. Get reports that correspond to the dataset.
   Given the dataset UUID and the dataset type, use them in the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain :9091/secondary-
analysis/datasets/{datasetType}/{datasetUUID}/reports
```

2.2 Diagram


GET DATASET REPORTS SEQUENCE DIAGRAM
SMRT API Client
Dataset Service
GET /secondary-analysis/dataset-types
Get corresponding dataset type string
GET /secondary-analysis/datasets/{datasetType}/{datasetUUID}/reports
SMRT API Client
Dataset Service

2.3 Example

To get reports associated with a subreadset with UUID = 146338e0-7ec2-4d2d-b938-11bce71b7ed1, perform the following steps:

Use the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/dataset-types
```

You see that the shortName of subreadsets is “subreads”. The desired endpoint is:

```
http://SMRTLinkServername.domain:9091/secondary-analysis/datasets/subreads/146338e0-
7ec2-4d2d-b938- 11bce71b7ed1/reports
```

Use the GET request with this endpoint to get reports that correspond to the subreadset with UUID = 146338e0-7ec2-4d2d-b938-11bce71b7ed1:

```
GET http://SMRTLinkServername.domain:9091/secondary-
analysis/datasets/subreads/146338e0-7ec2-4d2d-b938-11bce71b7ed1/reports
```

3.0 How to Obtain QC Reports for a Specific Run

3.1 Workflow

To obtain QC reports for a specific run, given the run Name, perform the following steps:

1. Get the list of all runs: Use the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domainsmrtlink:9091/smrt-link/run
```

2. In the response, perform a text search for the run Name: Find the object whose “name” field is equal to the run Name, and get the run UUID, which can be found in the “uniqueId” field.

3. Get all collections that belong to this run: Use the run UUID found in the previous step in the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domainsmrtlink:9091/smrt-link/runs/{runUUID}/collections
```

4. Take a collection UUID of one of collection objects received in the previous response. The collection UUIDs can be found in the “uniqueId” fields.
   - For **complete** collections, the collection UUID will be the same as the UUID of the subreadset for that collection.
   - Make sure that the collection whose “uniqueId” field you take has the field “status” set to “Complete”. This is because obtaining dataset reports based on the collection UUID as described below will **only** work if the collection is **complete**. If the collection is **not** complete, the subreadset does not exist yet.

5. Retrieve the QC reports that correspond to this collection: Use the collection UUID obtained in the previous step in the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-
analysis/datasets/subreads/{collectionUUID}/reports
```

6. Take a report UUID of one of the reports of the collection from the previous response. The report UUIDs can be found in the “uuid” fields.

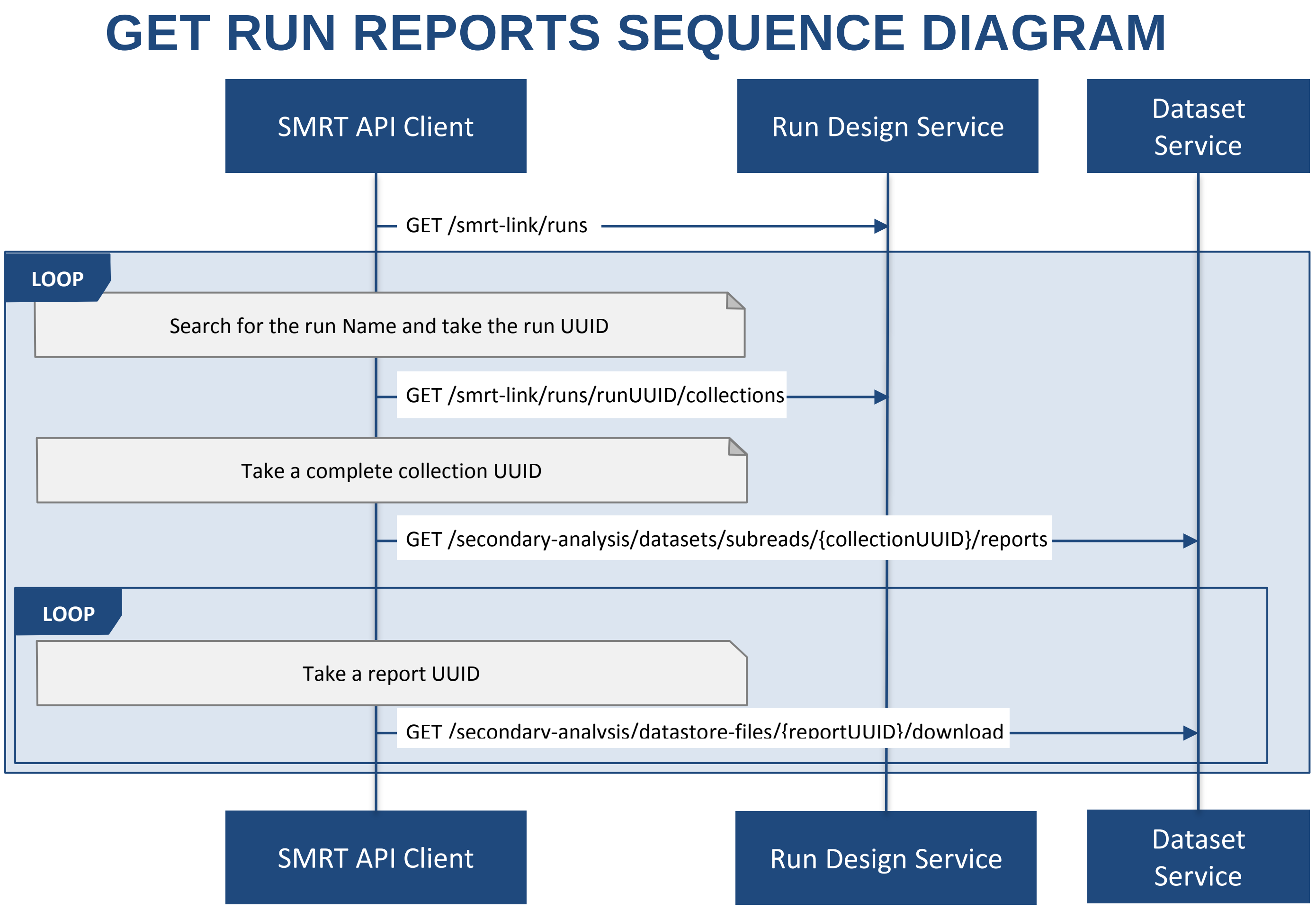
7. Download one of the reports associated with the collection: Use the report UUID in the GET request with the following endpoint:

```
GET http://smrtlink-release:9091/secondary-analysis/datastore-
files/{reportUUID}/download
```

8. Repeat Steps 6 - 7 to download all desired reports associated with that collection.

9. Repeat Steps 4 - 8 to download QC reports for all complete collections of that run.

3.2 Diagram


GET RUN REPORTS SEQUENCE DIAGRAM
SMRT API Client
Run Design Service
Dataset Service
GET /smrt-link/runs
LOOP
Search for the run Name and take the run UUID
GET /smrt-link/runs/runUUID/collections
Take a complete collection UUID
GET /secondary-analysis/datasets/subreads/{collectionUUID}/reports
LOOP
Take a report UUID
GET /secondary-analysis/datastore-files/{reportUUID}/download
SMRT API Client
Run Design Service
Dataset Service

3.3 Example

You view the Run QC page in the SMRT Link UI, and open the page of a run with status “Complete”. Take the run Name and look for the run UUID in the list of all runs, as described above.

Note: The run ID will also appear in the `{runUUID}` path parameter of the SMRT Link UI URL

```
http://SMRTLinkServername.domain:9091/#/run-qc/{runUUID}
```

So the shorter way would be to take the run UUID directly from the URL, such as

```
http://SMRTLinkServername.domain:9091/#/run-qc/d7b83cfc-91a6-4cea-8025-8bcc1f39e045
```

With this run UUID = d7b83cfc-91a6-4cea-8025-8bcc1f39e045, get all collections that belong to this run:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs/d7b83cfc-91a6-4cea-8025-
8bcc1f39e045/collections
```

Take a UUID of a completed collection, such as “uniqueId”: “59230aeb-a8e3-4b46-b1b1-24c782c158c1”. With this collection UUID, retrieve QC reports of the corresponding subreadset:

```
GET http://SMRTLinkServername.domain:9091/secondary-
analysis/datasets/subreads/59230aeb-a8e3-4b46-b1b1-24c782c158c1/reports
```

Take a UUID of some report, such as. “uuid”: “00c310ab-e989-4978-961e-c673b9a2b027”. With this report UUID, download the corresponding report file:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/datastore-files/00c310ab-
e989-4978-961e-c673b9a2b027/download
```

Repeat the last two API calls until you download all desired reports for all complete collections.

4.0 How to Obtain QC Reports for a Specific Collection

4.1 Workflow

For completed collections, the collection UUID will be the same as the UUID of the subreadset for that collection. To retrieve the QC reports of a completed collection, given the collection UUID, perform the following steps:

1. Get the QC reports that correspond to this collection: Use the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-
analysis/datasets/subreads/{collectionUUID}/reports
```

Note: Obtaining dataset reports based on the collection UUID as described above will only work if the collection is **complete**. If the collection is **not** complete, then the subreadset does not exist yet.

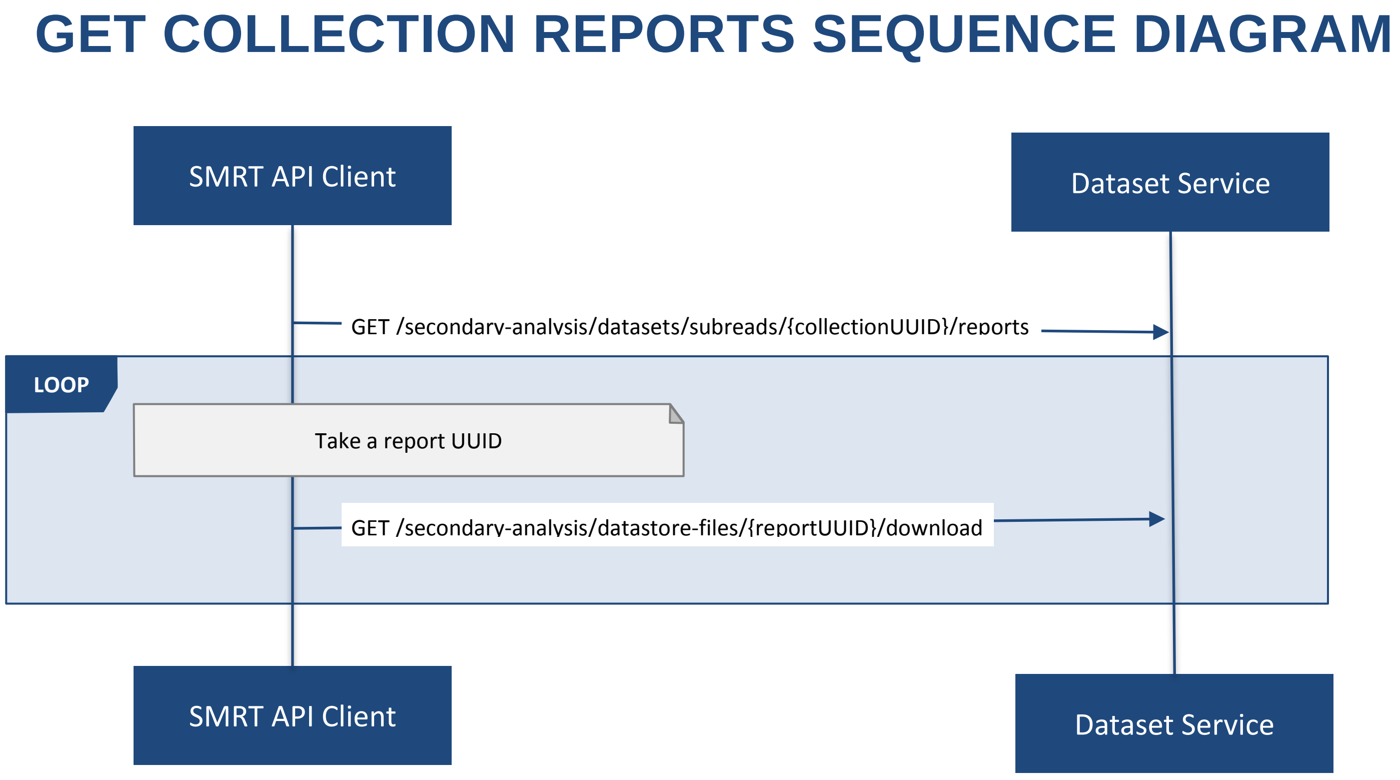
2. Take a report UUID of one of the reports of the collection from the previous response.
   The report UUIDs can be found in the "uuid" fields.

3. Download one of the reports of the collection: Use the report UUID in the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/datastore-
files/{reportUUID}/download
```

4. Repeat Steps 2 - 3 to download all desired reports of the collection.

4.2 Diagram


GET COLLECTION REPORTS SEQUENCE DIAGRAM
SMRT API Client
Dataset Service
GET /secondary-analysis/datasets/subreads/{collectionUUID}/reports
LOOP
Take a report UUID
GET /secondary-analysis/datastore-files/{reportUUID}/download
SMRT API Client
Dataset Service

4.3 Example

Suppose you have a complete collection with UUID = 59230aeb-a8e3-4b46-b1b1-24c782c158c1. Get all reports of the subreadset which corresponds to this collection:

```
GET http://SMRTLinkServername.domain:9091/secondary-
analysis/datasets/subreads/59230aeb-a8e3-4b46-b1b1-24c782c158c1/reports
```

Take the UUID of a desired report, such as "uuid": "00c310ab-e989-4978-961e-c673b9a2b027". With this report UUID, download the corresponding report file:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/datastore-files/00c310ab-
e989-4978-961e-c673b9a2b027/download
```

Repeat the last API call until you download all desired reports associated with this collection.

5.0 How to Obtain Recent Runs

5.1 Workflow

To obtain recent runs, perform the following steps:

1. Get the list of all runs: Use the GET request with the following endpoint:

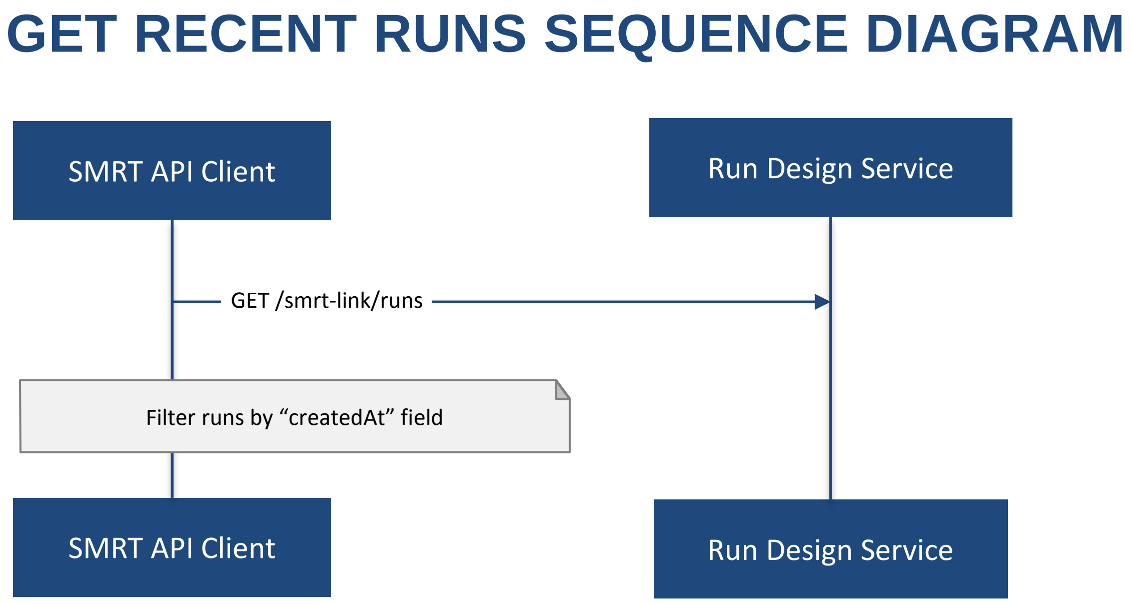
```
  GET http://SMRTLinkServername.domain:9091/smrt-link/runs
```

2. Filter the response based on the value of the "createdAt" field. For example:

```
"createdAt": "2016-12-13T19:11:54.086Z"
```

Note: You may also search runs based on specific criteria, such as reserved state, creator, or summary substring.

5.2 Diagram



5.3 Example

Suppose you want to find all runs created on or after 01.01.2017.

First, get the list of all runs:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs
```

The response will be an array of run objects, as in the following example:

```
[{
“name” : “2016-11-08_3150473_2kLambda_A12”,
“uniqueId” : “97286726-b243-45b3-82f7-8b5f58c56d53”,
“createdAt” : “2016-11-08T17:50:57.955Z”,
...
“summary” : “lambdaNEB”
}, {
“name” : “2017_01_24_A7_4kbSymAsym_DS_3150540”,
“uniqueId” : “abd8f5ec-a177-4d41-8556-81c5ffb6b0aa”,
“createdAt” : “2017-01-24T20:09:27.629Z”,
...
“summary” : “pBR322_InsertOnly”
}, {
“name” : “SMS_GoatVer_VVC034_3150433_2kLambda_400pm_SNR10.5”,
“uniqueId” : “b81de65a-8018-4843-9da7-ff2647a9d01e”,
“createdAt” : “2016-10-17T23:36:35.000Z”,
...
“summary” : “lambdaNEB”
}, {
```

```
"name" : "2017_01_21_A7_RC0_2.5-6kb_DS",
"uniqueId" : "5026afad-fbfa-407a-924b-f89dd019ca9f",
"createdAt" : "2017-01-21T00:21:52.534Z",
...
"summary" : "gencode_23_transcripts"
}
]
```

Now, search the above response for all run objects whose "createdAt" field starts with the "2017_01" substring. From the above example, you will get two runs that fit your criteria (that is, created on or after 01.01.2017):

- Run with "name" equal to "2017_01_24_A7_4kbSymAsym_DS_3150540",
- Run with "name" equal to "2017_01_21_A7_RC0_2.5-6kb_DS".

6.0 How to Set Up a Run Design

6.1 Workflow

To set up a run design, perform the following steps:

1. Prepare the Run Design information in an XML file. (The XML file should correspond to the PacBioDataModel.xsd schema.)

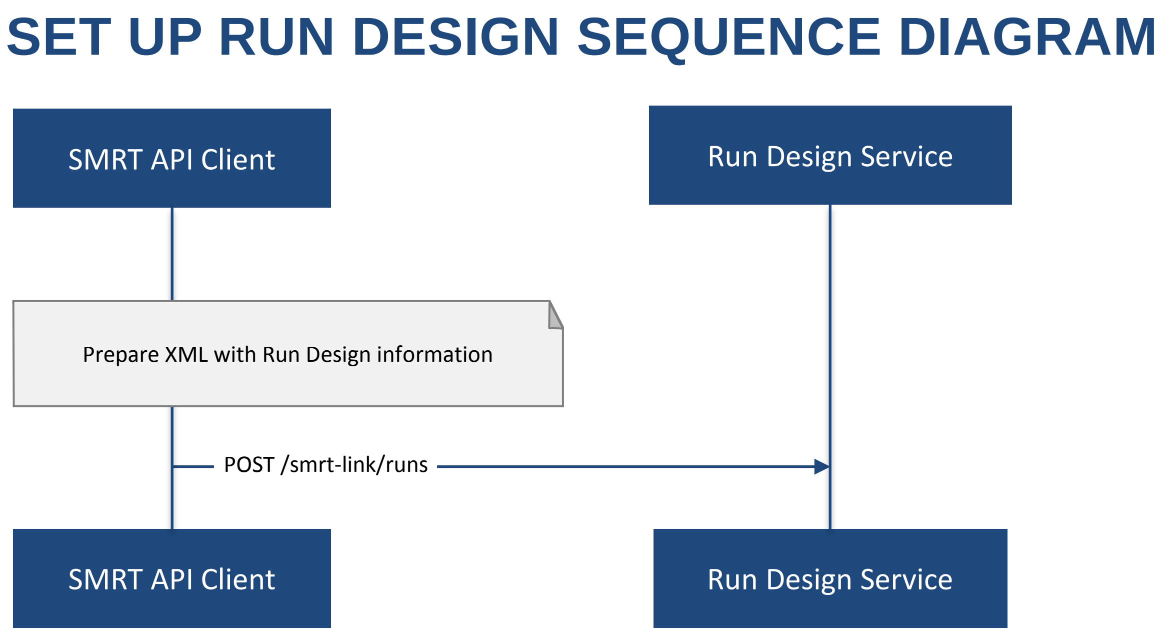
2. Create the run design: Use the POST request with the following endpoint:

```
POST http://SMRTLinkServername.domainsmrtlink:9091/smrt-link/runs
```

The payload (request body) for this POST request is a JSON with the following fields:

- **dataModel**: The serialized XML containing the Run Design information.
- **name**: The name of the run.
- **summary**: A short description of the run.

6.2 Diagram



6.3 Example

Create a run design using the following API call:

```
POST http://SMRTLinkServername.domain:9091/smrt-link/runs
```

Use the payload as in the following example:

```
{
"dataModel" : "<serialized Run Design XML file according to the PacBioDataModel.xsd
schema>",
     "name" : "Run_201601220309_D15",
     "summary" : "tkb_C5_circular_23x_I92782"
}
```

7.0 How to Monitor Run Progress

7.1 Workflow

Run progress can be monitored by looking at the completion status of each collection associated with that run. Perform the following steps:

1. If you do not have the run UUID, retrieve it as follows.
   a. Get the list of all runs, using the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs
```

   b. In the response, perform a text search for the run Name.

   Find the object whose "name" field is equal to the run Name, and get the run UUID, which can be found in the "uniqueId" field.

2. Once you have the run UUID, get all collections that belong to the run.

Use the run UUID in the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs/{runUUID}/collections
```

The response will contain the list of all collections of that run.

3. Monitor collection status to see when all collections are complete.
   Until all collections of the run have the field "status" set to "Complete", repeat the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs/{runUUID}/collections
```

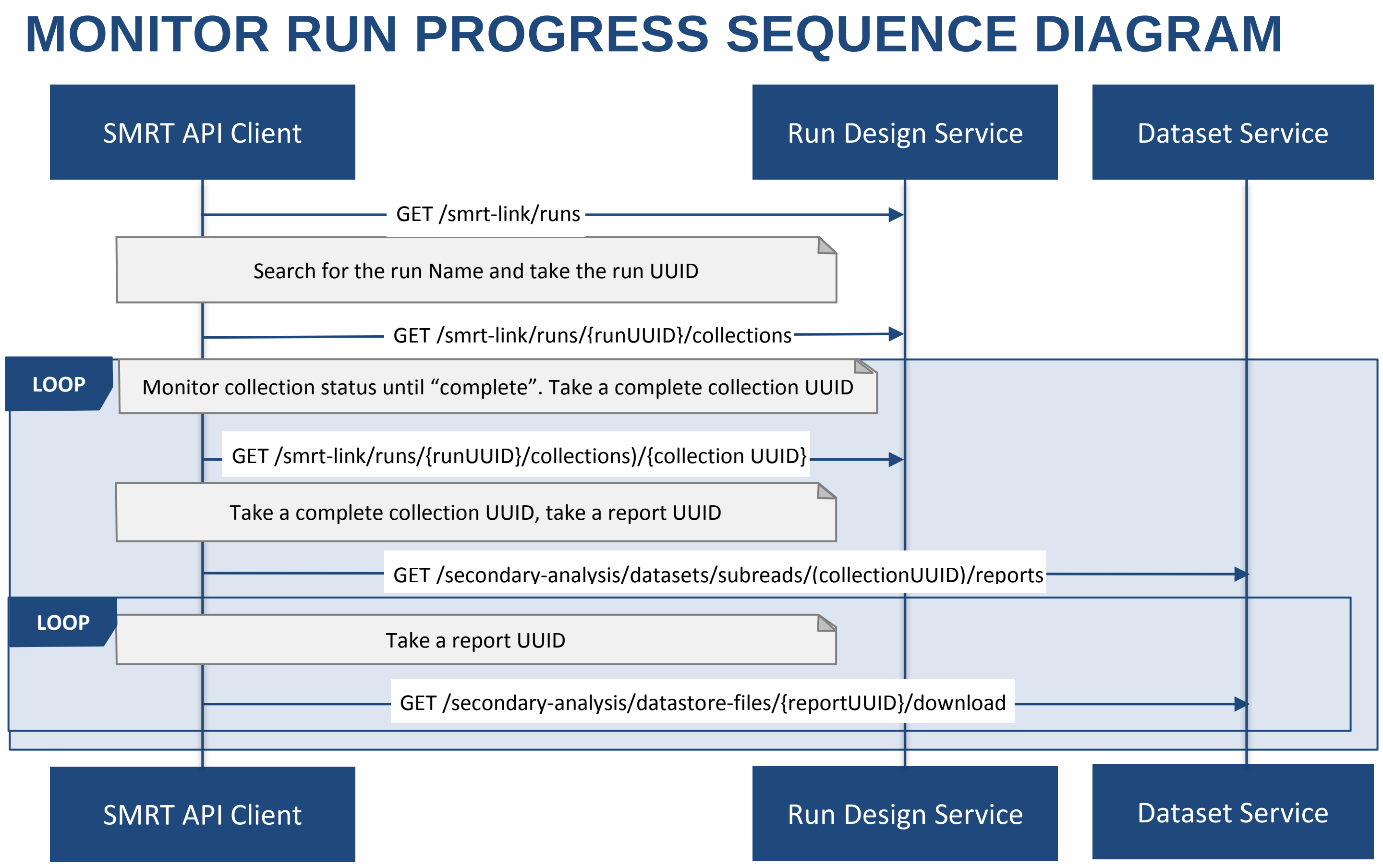
You may also monitor each collection individually.
Use the collection UUID in the GET request with the following endpoint:

```
GET http://SMRTLinkServername.domain:9091/smrt-
link/runs/{runUUID}/collections/{collectionUUID}
```

4. To monitor run progress using QC metrics as well, do this at the collection level, for each collection that belongs to this run. For instructions, see How to Obtain QC Reports for a Specific Collection.

The full set of QC metrics for a collection will **only** be available when the collection is **complete**. Monitor the completion status of each collection and, for each complete collection, check its QC metrics.
QC metrics of all collections that belong to the run will let you evaluate the overall success of the run.

7.2 Diagram


MONITOR RUN PROGRESS SEQUENCE DIAGRAM
SMRT API Client
Run Design Service
Dataset Service
GET /smrt-link/runs
Search for the run Name and take the run UUID
GET /smrt-link/runs/{runUUID}/collections
LOOP
Monitor collection status until "complete". Take a complete collection UUID
GET /smrt-link/runs/{runUUID}/collections)/{collection UUID}
Take a complete collection UUID, take a report UUID
GET /secondary-analysis/datasets/subreads/(collectionUUID)/reports
LOOP
Take a report UUID
GET /secondary-analysis/datastore-files/{reportUUID}/download
SMRT API Client
Run Design Service
Dataset Service

7.3 Example

If you want to monitor the run with Name = "54149_DryRun_2Cells_20161219", use the following steps:

1. Get the list of all runs: `GET http://SMRTLinkServername.domain:9091/smrt-link/runs`
   The response will be an array of run objects, as in the following example:

```
[{
"name" : "2016-11-08_3150473_2kLambda_A12",
"uniqueId" : "97286726-b243-45b3-82f7-8b5f58c56d53",
"createdAt" : "2016-11-08T17:50:57.955Z",
...
"summary" : "lambdaNEB"
}, {
...
}, {
"name" : "54149_DryRun_2Cells_20161219",
"uniqueId" : "798ff161-23ee-433a-bfd9-be8361b40f15",
"createdAt" : "2016-12-19T16:08:41.610Z",
...
"summary" : "DryRun_2Cells"
}, {
...
}, {
"name" : "2017_01_21_A7_RC0_2.5-6kb_DS",
"uniqueId" : "5026afad-fbfa-407a-924b-f89dd019ca9f",
"createdAt" : "2017-01-21T00:21:52.534Z",
...
```

```
  “summary” : “gencode_23_transcripts”
  }
]
```

2. Search the above response for the object with the "name" field equal to "54149_DryRun_2Cells_20161219".

   From the above example, you will get the run object with the "uniqueId" field equal to "798ff161-23ee-433a-bfd9-be8361b40f15".

3. With this run UUID = 798ff161-23ee-433a-bfd9-be8361b40f15, get all collections that belong to this run:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs/798ff161-23ee-433a-bfd9-be8361b40f15/collections
```

   The response will be an array of collection objects of this run, as in the following example:

```
  [{
             "name" : "DryRun_1stCell",
             "instrumentName" : "Sequel",
             "context" : "m54149_161219_161247",
             "well" : "A01",
             "status" : "Complete",
             "instrumentId" : "54149",
             "startedAt" : "2016-12-19T16:12:47.014Z",
             "uniqueId" : "7cf74b62-c6b8-431d-b8ae-7e28cfd8343b",
             "collectionPathUri" :
  "/pbi/collections/314/3140149/r54149_20161219_160902/1_A01",
             "runId" : "798ff161-23ee-433a-bfd9-be8361b40f15",
             "movieMinutes" : 120
       }, {
             "name" : "DryRun_2ndCell",
             "instrumentName" : "Sequel",
             "context" : "m54149_161219_184813",
             "well" : "B01",
             "status" : "Ready",
             "instrumentId" : "54149",
             "startedAt" : "2016-12-19T16:12:47.014Z",
             "uniqueId" : "08af5ab4-7cf4-4d13-9bcb-ae977d493f04",
             "collectionPathUri" :
  "/pbi/collections/314/3140149/r54149_20161219_160902/2_B01",
             "runId" : "798ff161-23ee-433a-bfd9-be8361b40f15",
             "movieMinutes" : 120
       }
  ]
```

In the above example, the first collection has “status” : “Complete”.
You can take its UUID, i.e. “uniqueId”: “7cf74b62-c6b8-431d-b8ae-7e28cfd8343b”, and get its QC metrics. For instructions, see How to Obtain QC Reports for a Specific Collection.

The second collection has “status” : “Ready”.
You can take its UUID, i.e. “uniqueId”: “08af5ab4-7cf4-4d13-9bcb-ae977d493f04”, and monitor its status until it becomes “Complete”; use the following API call:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs/798ff161-23ee-433a-bfd9-be8361b40f15/collections/08af5ab4-7cf4- 4d13-9bcb-ae977d493f04
```

Once this collection becomes complete, you can get its QC metrics as well. For instructions, see How to Obtain QC Reports for a Specific Collection.

8.0 How to Capture Run-Level Summary Metrics

Run-level summary metrics are captured in the QC reports. See the following sections:

- How to Obtain QC Reports for a Specific Run
- How to Obtain QC Reports for a Specific Collection

9.0 How to Set Up a Job on a Specific Collection

9.1 Workflow

To create a job using the SMRT Link Web Services API, use the POST request with the following endpoint:

```
POST http://SMRTLinkServername.domain:9091/secondary-analysis/job-manager/jobs/{jobTypeId}
```

The payload (request body) for this POST request is a JSON whose schema depends on the job type.

To specifically create a SMRT Analysis job, you need to create a job of type “pbsmrtpipe”.

You need to provide dataset IDs in the “entryPoints” array of the above payload.

To setup a job for a given collection, you need to specify the dataset ID for the subreadset of the collection in the request body of the Create Job by Type POST request.

Perform the following steps:

1. If you do not have the collection UUID, retrieve it as follows.

   To get the collection UUID starting from a run page in the SMRT Link Run QC UI, do the following:

   a. Get the run Name from the run page in the SMRT Link Run QC UI.
   b. Get the list of all runs, using the GET request with the following endpoint:

   ```
   GET http:/SMRTLinkServername.domain:9091/smrt-link/runs
   ```

   c. In the response, perform a text search for the run Name.

      Find the object whose “name” field is equal to the run Name, and get the run UUID, which can be found in the “uniqueId” field.

   d. Once you have the run UUID, get all collections that belong to this run. Use the run UUID in the GET request with the following endpoint:

   ```
   GET http://SMRTLinkServername.domain:9091/smrt-link/runs/{runUUID}/collections
   ```

   e. From here you can get the UUID of the collection. It can be found in the “uniqueId” field of the corresponding collection object from the previous response.

Note: Make sure that the collection whose “uniqueId” field you take has the field “status” set to “Complete”. This is because obtaining dataset ID based on the collection UUID as described below will **only** work if the collection is **complete**. If the collection is **not** complete, then the subreadset does not exist yet.

2. Find the dataset ID that corresponds to the collection UUID.
   For complete collections, the collection UUID will be the same as the UUID of the subreadset for that collection. Use the collection UUID in the GET request on the following endpoint to get the corresponding subreadset object:

   ```
   GET http://SMRTLinkServername.domain:9091/secondary-
   analysis/datasets/subreads/{collectionUUID}
   ```

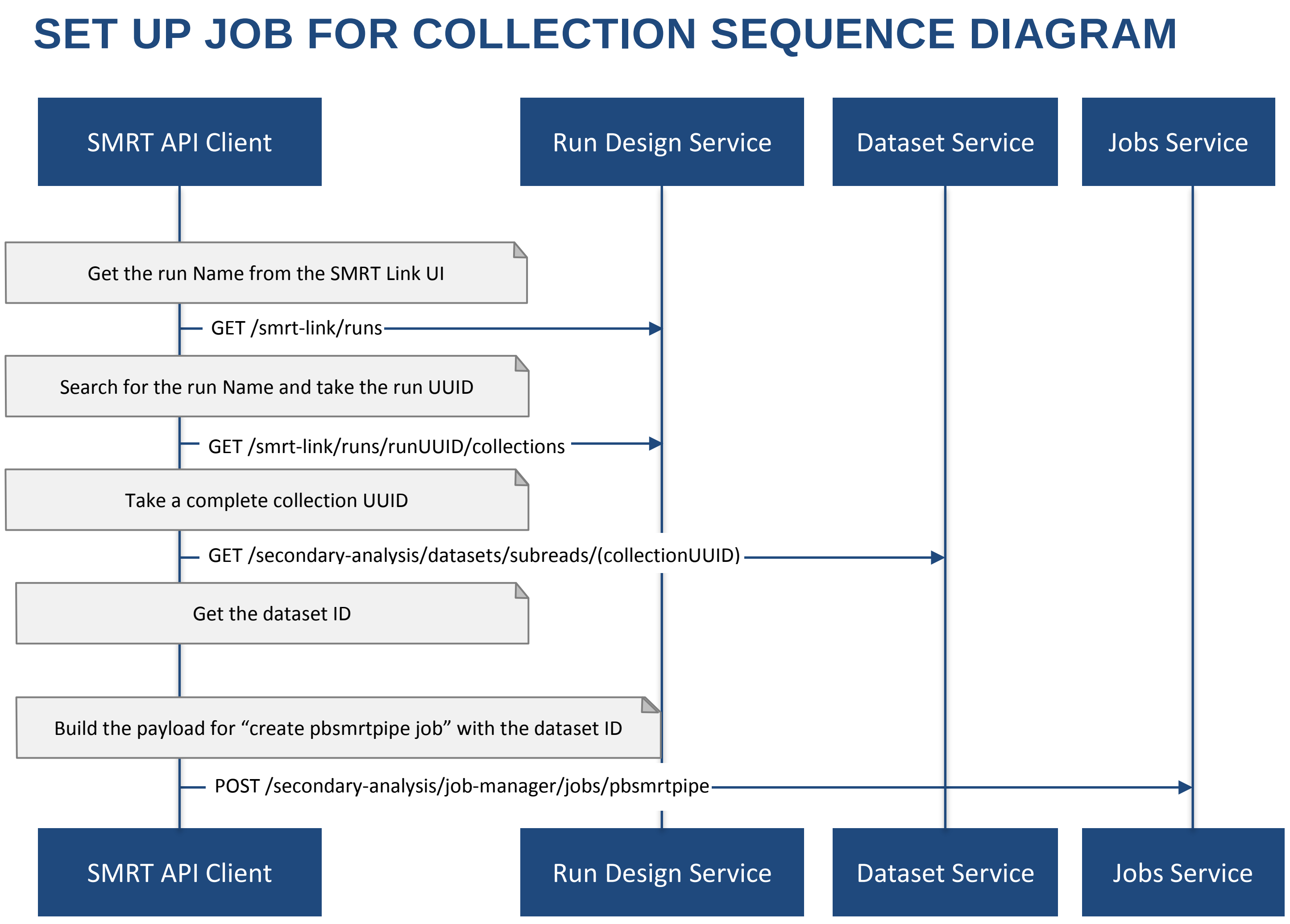
   Get the dataset ID from the “id” field of the response.

3. Build the request body with the dataset ID.
   Use the dataset ID in the payload.

4. Create a job of type “pbsmrtpipe”.
   Use the request body built in the previous step in the POST request with the following endpoint:

   ```
   POST http://SMRTLinkServername.domain:9091/secondary-analysis/job-
   manager/jobs/pbsmrtpipe
   ```

9.2 Diagram


SET UP JOB FOR COLLECTION SEQUENCE DIAGRAM
SMRT API Client
Run Design Service
Dataset Service
Jobs Service
Get the run Name from the SMRT Link UI
GET /smrt-link/runs
Search for the run Name and take the run UUID
GET /smrt-link/runs/runUUID/collections
Take a complete collection UUID
GET /secondary-analysis/datasets/subreads/(collectionUUID)
Get the dataset ID
Build the payload for “create pbsmrtpipe job” with the dataset ID
POST /secondary-analysis/job-manager/jobs/pbsmrtpipe
SMRT API Client
Run Design Service
Dataset Service
Jobs Service

9.3 Example

Suppose you want to setup a job for complete collections that belong to the run with Name = “54149_DryRun_2Cells_20161219”.

First, get the list of all runs: `GET http://SMRTLinkServername.domain:9091/smrt-link/runs`

The response will be an array of run objects, as in the following example:

```
 [{
            "name" : "2016-11-08_3150473_2kLambda_A12",
            "uniqueId" : "97286726-b243-45b3-82f7-8b5f58c56d53",
            "createdAt" : "2016-11-08T17:50:57.955Z",
            ...
            "summary" : "lambdaNEB"
     }, {
     ...
     }, {
            "name" : "54149_DryRun_2Cells_20161219",
            "uniqueId" : "798ff161-23ee-433a-bfd9-be8361b40f15",
            "createdAt" : "2016-12-19T16:08:41.610Z",
            ...
            "summary" : "DryRun_2Cells"
     }, {
     ...
     }, {
            "name" : "2017_01_21_A7_RC0_2.5-6kb_DS",
            "uniqueId" : "5026afad-fbfa-407a-924b-f89dd019ca9f",
            "createdAt" : "2017-01-21T00:21:52.534Z",
            ...
            "summary" : "gencode_23_transcripts"
     }
```

Now, search the above response for the object with the “name” field equal to “54149_DryRun_2Cells_20161219”.

From the above example, you will get the run object with the “uniqueId” field equal to “798ff161-23ee-433a-bfd9-be8361b40f15”.

With this run UUID = 798ff161-23ee-433a-bfd9-be8361b40f15, get all collections that belong to this run:

```
GET http://SMRTLinkServername.domain:9091/smrt-link/runs/798ff161-23ee-433a-bfd9-
be8361b40f15/collections
```

The response will be an array of collection objects of this run, as in the following example:

```
[{
            "name" : "DryRun_1stCell",
            "instrumentName" : "Sequel",
            "context" : "m54149_161219_161247",
            "well" : "A01",
            "status" : "Complete",
            "instrumentId" : "54149",
            "startedAt" : "2016-12-19T16:12:47.014Z",
            "uniqueId" : "7cf74b62-c6b8-431d-b8ae-7e28cfd8343b",
            "collectionPathUri" :
"/pbi/collections/314/3140149/r54149_20161219_160902/1_A01",
            "runId" : "798ff161-23ee-433a-bfd9-be8361b40f15",
            "movieMinutes" : 120
     }, {
            "name" : "DryRun_2ndCell",
            "instrumentName" : "Sequel",
            "context" : "m54149_161219_184813",
            "well" : "B01",
            "status" : "Ready",
            "instrumentId" : "54149",
            "startedAt" : "2016-12-19T16:12:47.014Z",
            "uniqueId" : "08af5ab4-7cf4-4d13-9bcb-ae977d493f04",
            "collectionPathUri" :
"/pbi/collections/314/3140149/r54149_20161219_160902/2_B01",
            "runId" : "798ff161-23ee-433a-bfd9-be8361b40f15",
            "movieMinutes" : 120
     }
]
```

In the above example, both collections of the run have “status” : “Complete”. Hence, the corresponding subreadsets should already exist, and can be retrieved as described below.

Take the UUID of the first collection,such as “uniqueId”: “7cf74b62-c6b8-431d-b8ae-7e28cfd8343b”, and get the corresponding subreadset object:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/datasets/subreads/7cf74b62-
c6b8-431d-b8ae-7e28cfd8343b
```

The response will be a subreadset object, as in the following example:

```
{
     “name” :
     “54149_DryRun_2Cells_20161219”, “uuid”
     : “7cf74b62-c6b8-431d-b8ae-
     7e28cfd8343b”, “id” : 5164,
     “createdAt” : “2016-12-19T19:20:46.968Z”,
     “path” :
     “/pbi/collections/314/3140149/r54149_20161219_160902/1_A01/m54149_161247.subreadset
     .xml”, “tags” : “subreadset”,
     “instrumentName” : “Sequel”,
     ...
     “wellExampleName” : “DryRun_1stCell”,
     “runName” :
     “54149_DryRun_2Cells_20161219”,
     “datasetType” :
     “PacBio.DataSet.SubreadSet”,
     “comments” : ” “
}
```

From the above response, take the value of the “id” field, which is 5164 in the above example. So dataset ID = 5164 will be the value for the first entry point for ‘pbsmrtpipe’ job.

Now take the UUID of the second collection, i.e. “uniqueId”: “08af5ab4-7cf4-4d13-9bcb-ae977d493f04”, and get the corresponding subreadset object:

```
GET http://SMRTLinkServername.domain:9091/secondary-analysis/datasets/subreads/08af5ab4-
7cf4-4d13-9bcb-ae977d493f04
```

The response will be a subreadset object, as in the following example:

```
{
     “name” : “54149_DryRun_2Cells_20161219”,
     “uuid” : “08af5ab4-7cf4-4d13-9bcb-
     ae977d493f04”, “id” : 5165,
     “createdAt” : “2016-12-19T21:57:11.173Z”,
     “path” :
     “/pbi/collections/314/3140149/r54149_20161219_160902/2_B01/m54149_184813.subreadset
     .xml”, “tags” : “subreadset”,
     “instrumentName” : “Sequel”,
     ...
     “wellExampleName” : “DryRun_2ndCell”,
     “runName” :
     “54149_DryRun_2Cells_20161219”,
     “datasetType” :
     “PacBio.DataSet.SubreadSet”,
     “comments” : ” “
}
```

From the response, again take the value of the “id” field, which is 5165 in the above example. So dataset ID = 5165 will be the value for the second entry point for ‘pbsmrtpipe’ job.

Build the request body for creating ‘pbsmrtpipe’ job. Use these two dataset IDs obtained above as values of the “datasetId” fields in the “entryPoints” array. For example:

```
{
"name" : "A4_All4mer_1hr_launchChem",
"entryPoints" :
[{
             "entryId" : "eid_subread",
             "fileTypeId" : "PacBio.DataSet.SubreadSet",
             "datasetId" : 5164
      }, {
             "entryId" : "eid_subread2",
             "fileTypeId" : "PacBio.DataSet.SubreadSet",
             "datasetId" : 5165
      }
],
"workflowOptions" : [],
"taskOptions" :
[{
      "optionId" : "genomic_consensus.task_options.algorithm",
             "value" : "quiver",
             "optionTypeId" : "pbsmrtpipe.option_types.string"
      }, {
             "optionId" : "genomic_consensus.task_options.diploid",
             "value" : false,
             "optionTypeId" : "pbsmrtpipe.option_types.boolean"
      }
],
"pipelineId" : "pbsmrtpipe.pipelines.sa3_resequencing"
}
```

Now create a job of type “pbsmrtpipe”. Use the request body built above in the following API call:

```
POST http://SMRTLinkServername.domain:9091/secondary-analysis/job-manager/jobs/pbsmrtpipe
```

Verify that the job was created successfully. The return HTTP status should be **201 Created**.

10.0 How to Delete a Job

10.1 Workflow

To delete a job, you need to create another job of type “delete-job”, and pass the UUID of the job to delete in the payload (a.k.a. request body).

Perform the following steps:

1. Build the payload for the POST request as a JSON with the following fields:

- **jobId**: The UUID of the job to be deleted.
- **removeFiles**: A boolean flag specifying whether to remove files associated with the job being deleted.
- **dryRun**: A boolean flag for checking whether it is safe to delete the job prior to actually deleting it.

Note: If you want to make sure that it is safe to delete the job (there is no other piece of data dependent on the job being deleted), then first set the the “dryRun” field to ‘true’ and perform the API call described in Step 2 below. If the call succeeds, meaning that the job can be safely deleted, set the “dryRun” field to ‘false’ and repeat the same API call again, as described in Step 3 below.

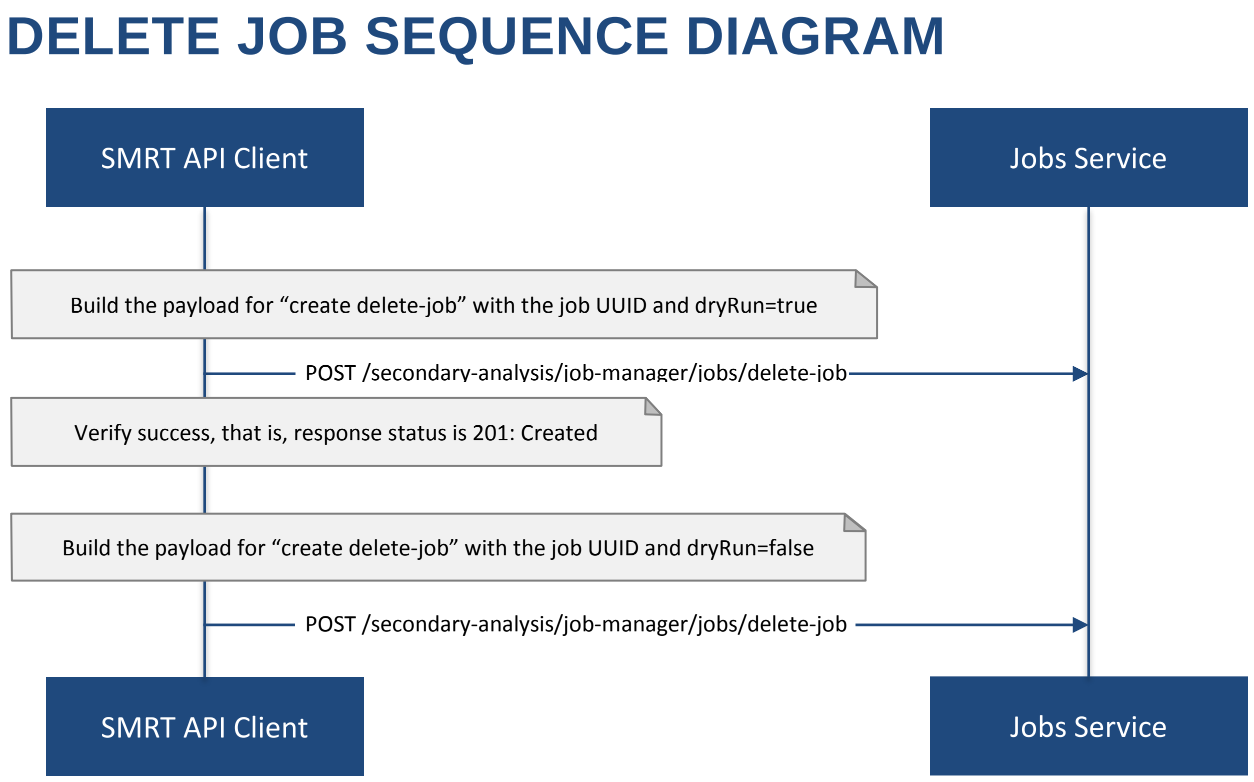
2. Check whether the job can be deleted, without actually changing anything in the database or on disk. Create a job of type “delete-job” with the payload which has dryRun = true; use the POST request with the following endpoint:

```
POST http://SMRTLinkServername.domain:9091/secondary-analysis/job-manager/jobs/delete-
job
```

3. If the previous API call succeeded, that is, the job may be safely deleted, then proceed with actually deleting the job. Create a job of type “delete-job” with the payload which has dryRun = false; use the POST request with the following endpoint:

```
POST http://SMRTLinkServername.domain:9091/secondary-analysis/job-manager/jobs/delete-
job
```

10.2 Diagram


DELETE JOB SEQUENCE DIAGRAM
SMRT API Client
Jobs Service
Build the payload for "create delete-job" with the job UUID and dryRun=true
POST /secondary-analysis/job-manager/jobs/delete-job
Verify success, that is, response status is 201: Created
Build the payload for "create delete-job" with the job UUID and dryRun=false
POST /secondary-analysis/job-manager/jobs/delete-job
SMRT API Client
Jobs Service

10.3 Example

Suppose you want to delete the job with UUID = 13957a79-1bbb-44ea-83f3-6c0595bf0d42.

Define the payload as in the following example, and set the "dryRun" field in it to 'true':

```
{
     "jobId" : "13957a79-1bbb-44ea-83f3-
     6c0595bf0d42", "removeFiles" : true,
     "dryRun" : true
}
```

Create a job of type "delete-job", using the above payload in the following POST request:

```
POST http://SMRTLinkServername.domain:9091/secondary-analysis/job-manager/jobs/delete-job
```

Verify that the response status is **201: Created**.
Also notice that the response body contains JSON corresponding to the job to be deleted, as in the following example:

```
{
"name" : "Job merge-datasets",
"uuid" : "13957a79-1bbb-44ea-83f3-6c0595bf0d42",
"jobTypeId" : "merge-datasets",
"id" : 53,
"createdAt" : "2016-01-29T00:09:58.462Z",
...
"comment" : "Merging Datasets MergeDataSetOptions(PacBio.DataSet.SubreadSet, Auto-
merged subreads @1454026198403)"
}
```

Define the payload as in the following example, and this time set the “dryRun” field to ‘false’, to actually delete the job:

```
{
“jobId” : “13957a79-1bbb-44ea-83f3-6c0595bf0d42”, “removeFiles” : true,
“dryRun” : false
}
```

Create a job of type “delete-job”, using the above payload in the following POST request:

```
POST http://SMRTLinkServername.domain:9091/secondary-analysis/job-
manager/jobs/delete-job
```

Verify that the response status is **201: Created**. Notice that this time the response body contains JSON corresponding to the job of type “delete-job”, as in the following example:

```
{
     “name” : “Job delete-job”,
     “uuid” : “1f60c976-e426-43b5-
     8ced-f8139de6ceff”,
     “jobTypeId” : “delete-job”,
     “id” : 7666,
     “createdAt” : “2017-03-09T11:51:38.828-08:00”,
     ...
     “comment” : “Deleting job 13957a79-1bbb-44ea-83f3-6c0595bf0d42”
}
```

11.0 How to Set Up an Analysis Job for a Specific Pipeline

11.1 Workflow

To create an analysis job for a specific pipeline, you need to create a job of type “pbsmrtpipe” with the payload based on the template of the desired pipeline. Perform the following steps:

1. Get the list of all pipeline templates used for creating analysis jobs:

```
GET http://smrtlink-release:9091/secondary-analysis/resolved-pipeline-templates
```

2. In the response, search for the name of the specific pipeline that you want to set up. Once the desired template is found, note the values of the pipeline “id” and “entryPoints” elements of that template.

3. Get the datasets list that corresponds to the type specified in the first element of “entryPoints” array. For example, for the type “fileTypeId” : “PacBio.DataSet.SubreadSet”, get the list of “subreads” datasets:

```
GET http://smrtlink-release:9091/secondary-analysis/datasets/subreads
```

4. Repeat step 3. for the dataset types specified in the rest of elements of “entryPoints” array.

5. From the lists of datasets brought on steps 3. and 4, select IDs of the datasets that you want to use as entry points for the pipeline you are about to set up.

6. Build the request body for creating a job of type “pbsmrtpipe”.
   - Use the pipeline “id” found on step 2 as the value for “pipelineId” element.
   - Use dataset types of “entryPoints” array found on step 2 and corresponding dataset IDs found on step 5 as the values for elements of “entryPoints” array.
   - Note that “taskOptions” array is optional and may be completely empty in the request body.

7. Create a job of type “pbsmrtpipe”.
   Use the request body built in the previous step in the POST request with the following endpoint:

```
POST http://smrtlink-release:9091/secondary-analysis/job-manager/jobs/pbsmrtpipe
```

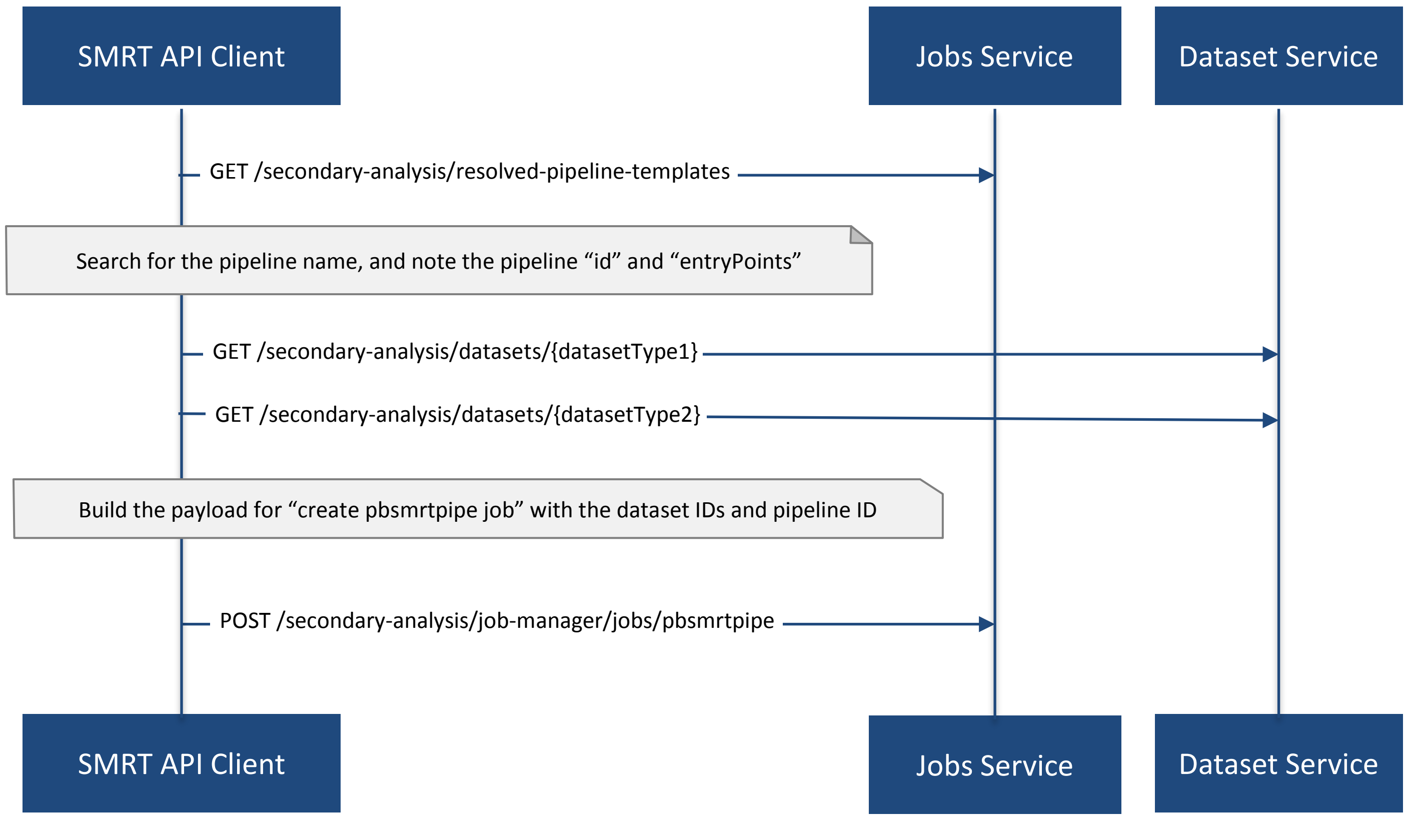
8. You may monitor the state of the job created on step 7 with the use of the following request:

```
GET http://smrtlink-release:9091/secondary-analysis/job-
manager/jobs/pbsmrtpipe/{jobID}/events
```

   where jobID is equal to the value received in “id” element of the response on step 7.

11.2 Diagram

SETUP ANALYSIS JOB FOR PIPELINE SEQUENCE DIAGRAM


SMRT API Client
Jobs Service
Dataset Service
GET /secondary-analysis/resolved-pipeline-templates
Search for the pipeline name, and note the pipeline “id” and “entryPoints”
GET /secondary-analysis/datasets/{datasetType1}
GET /secondary-analysis/datasets/{datasetType2}
Build the payload for “create pbsmrtpipe job” with the dataset IDs and pipeline ID
POST /secondary-analysis/job-manager/jobs/pbsmrtpipe
SMRT API Client
Jobs Service
Dataset Service

11.3 Example

Suppose you want to setup an analysis job for Resequencing pipeline.

First, get the list of all pipeline templates used for creating analysis jobs:

```
GET http://smrtlink-release:9091/secondary-analysis/resolved-pipeline-templates
```

The response will be an array of pipeline template objects. In this response, do the search for the entry with “name” : “Resequencing”. The entry may look as in the following example:

```
{
    “name” : “Resequencing”,
    “id” : “pbsmrtpipe.pipelines.sa3_ds_resequencing_fat”,
    “description” : “Full Resequencing Pipeline - Blasr mapping and Genomic
Consensus.”,
    “version” : “0.1.0”,
    “entryPoints” : [{
        “entryId” : “eid_subread”,
        “fileTypeId” : “PacBio.DataSet.SubreadSet”,
        “name” : “Entry Name: PacBio.DataSet.SubreadSet”
    }, {
        “entryId” : “eid_ref_dataset”,
        “fileTypeId” : “PacBio.DataSet.ReferenceSet”,
        “name” : “Entry Name: PacBio.DataSet.ReferenceSet”
      }
    ],
    “tags” : [
       “consensus”,
       “reports”
    ],
    “taskOptions” : [{
        “name” : “Diploid mode (experimental)”,
        “description” : “Enable detection of heterozygous variants (experimental)”,
        “id” : “genomic_consensus.task_options.diploid”,
        “optionTypeId” : “boolean”,
        “default” : false
    }, {
        “name” : “Purpose”,
        “description” : “Run mode (‘variants’ or ‘coverage’)”,
        “id” : “genomic_consensus.task_options.gff2bed_purpose”,
        “optionTypeId” : “string”,
        “default” : “variants”
       }
    ]
}
```

In the above entry, take the value of the pipeline “id” : “pbsmrtpipe.pipelines.sa3_ds_resequencing_fat”. Also, take the dataset types of “entryPoints” elements: “fileTypeId” : “PacBio.DataSet.SubreadSet” and “fileTypeId” : “PacBio.DataSet.ReferenceSet”.

Now, get the lists of the datasets that correspond to the types specified in the elements of the “entryPoints” array.

In particular, for the type “fileTypeId” : “PacBio.DataSet.SubreadSet”, get the list of “subreads” datasets:

```
GET http://smrtlink-release:9091/secondary-analysis/datasets/subreads
```

And for the type “fileTypeId” : “PacBio.DataSet.ReferenceSet”, get the list of “references” datasets:

```
GET http://smrtlink-release:9091/secondary-analysis/datasets/references
```

From the above lists of datasets, select IDs of the datasets that you want to use as entry points for the Resequencing pipeline you are about to setup.

For example, take the dataset with “id”: 18 from the “subreads” list and the dataset with “id”: 2 from the “references” list.

Build the request body for creating ‘pbsmrtpipe’ job for Resequencing pipeline.
Use the pipeline “id” obtained above as the value for “pipelineId” element.
Use these two dataset IDs obtained above as values of the “datasetId” fields in the “entryPoints” array. For example:

```
{
“pipelineId” : “pbsmrtpipe.pipelines.sa3_ds_resequencing_fat”,
“entryPoints” :
[{
“entryId” : “eid_subread”,
“fileTypeId” : “PacBio.DataSet.SubreadSet”,
“datasetId” : 18
}, {
“entryId” : “eid_ref_dataset”,
“fileTypeId” : “PacBio.DataSet.ReferenceSet”,
“datasetId” : 2
}
],
“taskOptions” : []
}
```

Now create a job of type “pbsmrtpipe”.
Use the request body built above in the following API call:

```
POST http://smrtlink-release:9091/secondary-analysis/job-manager/jobs/pbsmrtpipe
```

Verify that the job was created successfully. The return HTTP status should be **201 Created**.

P/N 101-430-800 Version 01 (February 2018)