SMRT® Link v8.0 Migration Guide

Quick Reference Card

Overview

Important - SMRT Link v7.0.0 (and earlier) Users: SMRT Link v8.0 uses the Broad Institute's Cromwell workflow engine, and replaces **all** `pbsmrtpipe` pipelines with Cromwell workflows. (See https://cromwell.readthedocs.io/en/develop/ for details).

This document summarizes the changes in SMRT Link v8.0 in relation to prior SMRT Link versions and provides useful information for planning an upgrade to SMRT Link v8.0.

Cromwell is a scientific workflow engine designed for scalability and specificity. It has become an industry tool used in many scientific environments. Its advantages include better utilization and management of computational resources, support for 3rd party tools and Cloud environments.

Using Cromwell in SMRT Link provides the following capabilities:

- Automatic retry in case of failed analysis failure.
- Better parallelization of analysis tasks resulting in faster time to results.
- Enables consolidation of analysis pipelines

General Changes

Common for all SMRT Link users:

- Directory structure, logs, and primary outputs from SMRT Analysis.

SMRT Link GUI users:

- Functional changes are minimized. Most of the same applications continue to appear in the SMRT Analysis module, although their back-end implementations are completely different.
- Base modification analysis and Iso-Seq® analysis are now each single analysis applications. You can run Motif Analysis as part of Base modification application, and mapping or clustering as part of the Iso-Seq Analysis application.
- Failed analyses can be restarted using a Restart button on the results page. Analysis will be re-executed from the point of failure.

  Note: Analyses ran on SMRT Link v7.0 or earlier can not be automatically re-run on SMRT Link v8.0. A new analysis will need to be created and run.

SMRT Link Command Line Users:

- pbsmrtpipe is replaced by pbcromwell, and all pipeline and task option IDs have changed.
- Old pbsmrtpipe presets XML/JSON are not supported and use native Cromwell input JSON format (simple key:value dictionary).
- The outputs/ directory includes symlinks for user convenience.
- Cromwell workflows may be run on the command-line by submitting a job with `pbservice` (using the same syntax as `pbsmrtpipe` jobs in previous versions), or as pure command-line jobs using the `pbcromwell` wrapper command.
- Existing automation scripts and configuration files will need to be updated or replaced to conform to new file formats and syntax. Consult documentation for the `pbcromwell` tool (SMRT Tools Reference Guide.)
- Use `pbcromwell configure` to generate a Cromwell config file for adjusting HPC settings.

SMRT Link API Users:

- Pipelines and task option IDs have changed.
- Analysis jobs now have the jobtype "analysis" (replacing "pbsmrtpipe"), i.e. the endpoint is /smrt-link/job-manager/jobs/analysis. (Old pbsmrtpipe jobs can also be retrieved from this endpoint.)
- pbservice run-pipeline works as before, but --task-options has been replaced by --task-option (which may be repeated as needed).

SMRT Link Site Administrators:

- When upgrading from a previous version of SMRT Link, compute configuration does not need to change - existing settings will be propagated and respected.
- Multiple compute configurations continue to be supported as before.
- Many of the compute settings specific to pbsmrtpipe, such as MAX_TOTAL_NPROC and MAX_NWORKERS, no longer apply to Cromwell. Instead, the Cromwell configuration throttles the number of simultaneous tasks globally.
- If your computational cluster has lots of processors, you can increase maxchunks to 96 for faster analysis job execution.
- Most output files are stored in a separate cromwell-executions directory in jobs_root, outside the hierarchy of job output directories (but also symlinked from individual jobs).

Additional Details for SMRT Link CMD Users

Applications:

From the `pbservice` CLI and the REST API, the applications can be invoked similarly to `pbsmrtpipe` pipelines. The table below shows how each analysis application in the GUI maps to a workflow ID that command-line tools will recognize.

Application	Cromwell Workflow
Assembly (HGAP4)	`cromwell.workflows.pb_hgap4`
Base Modification Detection	`cromwell.workflows.pb_basemods`
Base Modification Detection and Motif Analysis (v7.0.0)	`cromwell.workflows.pb_basemods with find_motifs=true`
Resequencing	`cromwell.workflows.pb_resequencing`
CCS with Mapping	`cromwell.workflows.pb_ccs_mapping`
Circular Consensus Sequencing (CCS)	`cromwell.workflows.pb_ccs`
Demultiplex Barcodes (Subreads)	`cromwell.workflows.pb_demux_subreads`
Demultiplex Barcodes (CCS)	`cromwell.workflows.pb_demux_ccs`
Site Acceptance Test (SAT)	`cromwell.workflows.pb_sat`
Iso-Seq	`cromwell.workflows.pb_isoseq3 (without ReferenceSet)`
Iso-Seq with Mapping (v7.0.0 and earlier)	`cromwell.workflows.pb_isoseq3 with optional ReferenceSet input`
Iso-Seq Classify Only (v7.0.0 and earlier)	`cromwell.workflows.pb_isoseq3 with run_clustering=false`
Iso-Seq (CCS) (New in v8.0.0)	`cromwell.workflows.pb_isoseq3_ccsonly`
Mapping (CCS)	`cromwell.workflows.pb_align_ccs`
Subread Mapping	`cromwell.workflows.pb_align_subreads`
Long Amplicon Analysis (LAA)	`cromwell.workflows.pb_laa`
Structural Variant Calling	`cromwell.workflows.pb_sv_clr or cromwell.workflows.pb_sv_ccs`
Minor Variants	`cromwell.workflows.pb_mv_ccs`
Microbial Assembly (New in v8.0.0)	`cromwell.workflows.pb_assembly_microbial`

More information can be obtained on any application by running `pbservice:` (**Note**: An identical command is also available in `pbcromwell`.)

```
$ pbservice show-workflows pb_resequencing
Pipeline ID: cromwell.workflows.pb_resequencing
Name: Resequencing
Entry Points:
  eid_subread: PacBio.DataSet.SubreadSet
  eid_ref_dataset: PacBio.DataSet.ReferenceSet
Task Options:
  consensus_algorithm: arrow
  dataset_filters:
  mapping_biosample_name:
  mapping_min_concordance: 70.0
  mapping_min_length: 50
  downsample_factor: 0
  consolidate_bam: true
  internal_max_nchunks_mapping: 100
  internal_max_nchunks_consensus: 40
```

Several Details to Note:

- `pbservice` now assumes a default prefix of `cromwell.workflows`, instead of `pbsmrtpipe.pipelines`.
- The prefixes for task options have been removed. The IDs were kept similar to equivalent `pbsmrtpipe` options where possible, for example `pbcoretools.task_options.downsample_factor` becomes `downsample_factor`.
- The `pbservice` rules for entry points are the same as for `pbsmrtpipe`, except that the Iso-Seq workflow accepts an optional genomic reference and runs mapping **only** if the reference is provided (instead of being a separate application).
- If you do **not** want call caching, add the argument `--no-cache` when running `pbservice run-pipeline`.
- `pbsmrtpipe` preset XMLs are **obsolete** and no longer supported; please use JSON files in the future.
- The `pbservice` CLI was changed slightly for more flexible task option input:

```
# old usage:
$ pbservice run-pipeline sa3_sat ... --task-options
pbcoretools.task_options.downsample_factor=10,pbcoretools.tasks.dataset_filters="rq >= 0.75"

# new usage:
$ pbservice run-pipeline pb_sat ... --task-option downsample_factor=10 --task-option
dataset_filters="rq >= 0.75"
```

In general, performance results should be nearly identical to the old applications, with small variations due to chunking behavior. For most analyses, the Cromwell workflow should be at least as fast as `pbsmrtpipe`, and in most cases significantly faster.

Following are popular task options and their equivalents in the new workflows (the default values should be the same in most cases):

pbsmrtpipe Option	**Cromwell Option**
`pbcoretools.task_options.downsample_factor`	`downsample_factor`
`pbcoretools.task_options.other_filters`	`dataset_filters`
`pbcoretools.task_options.consolidate_aligned_bam`	`consolidate_aligned_bam`
`pbcoretools.task_options.output_unaligned_bam`	`extract_unmapped_bam`
`pbmm2_align.task_options.minalnlength`	`mapping_min_length`
`pbmm2_align.task_options.min_perc_concordance`	`mapping_min_concordance`
`pbccs.task_options.polish`	`ccs_polish`
`pbccs.task_options.min_predicted_accuracy`	`ccs_min_predicted_accuracy`
`pbccs.task_options.min_passes`	`ccs_min_passes`
`falcon_ns.task_options.HGAP_GenomeLength_str`	`HGAP_GenomeLength_str`
`falcon_ns.task_options.HGAP_SeedCoverage_str`	`HGAP_SeedCoverage_str`
`falcon_ns.task_options.HGAP_SeedLengthCutoff_str`	`HGAP_SeedLengthCutoff_str`
`falcon_ns.task_options.HGAP_AggressiveAsm_bool`	`HGAP_AggressiveAsm_bool`
`falcon_ns.task_options.HGAP_FalconAdvanced_str`	`HGAP_FalconAdvanced_str`

Call Caching and Restarts:

All analysis jobs will write results to the cache database unless specifically told not to (currently the pbservice option --no-cache is the only method to turn this off). However, for the v8.0 release, reading from the cache is only enabled for "restarted" jobs - which are entirely new jobs that can retrieve existing task outputs.

Nothing significant needs to be done to enable this. Note that the call cache will only be used if the **Restart** button is clicked on the SMRT Link GUI. Re-running an analysis from scratch, or using "copy from" when creating the analysis, will run all analysis tasks from the beginning.

Compute Settings:

Unlike `pbsmrtpipe`, the Cromwell engine itself scales very efficiently with the number of chunks, so there is no system-based limit on `max_nchunks`. However, I/O bottlenecks limit the usefulness of very large numbers of chunks, so the default behavior in the SMRT Link server is to use the **same** `nchunks/nproc` settings as previous versions. However, our workflows have been tested up to at least 100 chunks, and users should experiment with their own environment as individual systems and scientific applications will vary in performance.

SMRT Link v8.0 uses a very similar configuration mechanism to v7.0.0, including support for multiple compute configurations. As a result, users who primarily run analyses on the server do not need to change anything. The `pbcromwell` tool does **not** use `pbsmrtpipe`-format preset JSON or XML; instead, `nproc` and `max_nchunks` are command-line arguments, and back-end settings can be manipulated by creating a Java configuration file that is passed to Cromwell (see below for an example of how to generate such a file). This gives users maximum flexibility and control while keeping the interface as close as possible to the Cromwell documentation.

Note: `nproc` and `max_nchunks` are **not** properties of the workflow engine - these are specific to PacBio applications, although these propagate `nproc` down to the back-end submission scripts, such as `qsub` and analogous commands.

```
$ pbcromwell configure --output-file cromwell.conf --jms-job-limit 100
$ pbcromwell run pb_ccs -e <SUBREADS> --config cromwell.conf --nproc 8 --backend SGE
```

Command Line Execution:

The command `pbcromwell run` is the replacement for the obsolete command `pbsmrtpipe run-pipeline`. The command-line options are mostly new and different, but a few similarities remain, such as the use of `-e <FILENAME>` to denote a dataset XML entry point. A detailed description of program behavior is available by running `pbcromwell run --help`.

The tool will always run in a new output directory and will exit with an error if the directory exists; use `--overwrite` to delete the directory and create a new one. Like SMRT Link jobs, output files will be symlinked to a subdirectory named `outputs`. All other output files are in native Cromwell formats - the `pbcromwell` code is just a thin layer around the workflow engine and produces very little output of its own other than intermediate files passed to Cromwell.

Back-end Changes:

Datastore: The format is unchanged, but there are some key differences:

- Most sourceIds have changed, typically to `pb_resequencing.mapped` or similar. As these are defined at the workflow level rather than the task level, they should be more stable than `pbsmrtpipe` output file sourceIds were in the past.

- The datastore JSON format was kept as it is integral to SMRT Link and contains useful metadata, but it is significantly smaller now and the file IDs are unambiguous. Users may prefer to look at `metadata-summary.json` (also at the top level of the job directory). This is a subset of what Cromwell's own REST API tells SMRT Link about the final state of the workflow, and includes a simple dictionary of output files:

```
    {
  "workflowName": "pb_demux_subreads_auto",
  "outputs": {
    "pb_demux_subreads_auto.barcoded_reports_datastore": "/pbi/dept/secondary/siv/smrtlink/smrtlink-bihourly/jobs-root/cromwell-executions/pb_demux_subreads_auto/4e85a878-6316-40f5-82e4-b44c46691768/call-demultiplex_barcodes/demultiplex_barcodes/64d1bebf-3811-4c21-bc1e-cf0c43f65fe8/call-barcode_report/execution/barcoded_reports.datastore.json",
    "pb_demux_subreads_auto.report_barcodes": "/pbi/dept/secondary/siv/smrtlink/smrtlink-bihourly/jobs-root/cromwell-executions/pb_demux_subreads_auto/4e85a878-6316-40f5-82e4-b44c46691768/call-demultiplex_barcodes/demultiplex_barcodes/64d1bebf-3811-4c21-bc1e-cf0c43f65fe8/call-barcode_report/execution/barcodes.report.json",
    "pb_demux_subreads_auto.unbarcoded": "/pbi/dept/secondary/siv/smrtlink/smrtlink-bihourly/jobs-root/cromwell-executions/pb_demux_subreads_auto/4e85a878-6316-40f5-82e4-b44c46691768/call-demultiplex_barcodes/demultiplex_barcodes/64d1bebf-3811-4c21-bc1e-cf0c43f65fe8/call-gather_unbarcoded/execution/merged.bam",
    "pb_demux_subreads_auto.barcoded_reads": "/pbi/dept/secondary/siv/smrtlink/smrtlink-bihourly/jobs-root/cromwell-executions/pb_demux_subreads_auto/4e85a878-6316-40f5-82e4-b44c46691768/call-demultiplex_barcodes/demultiplex_barcodes/64d1bebf-3811-4c21-bc1e-cf0c43f65fe8/call-update_barcoded_sample_metadata/execution/metadata_updated.datastore.json",
    "pb_demux_subreads_auto.summary_csv": "/pbi/dept/secondary/siv/smrtlink/smrtlink-bihourly/jobs-root/cromwell-executions/pb_demux_subreads_auto/4e85a878-6316-40f5-82e4-b44c46691768/call-demultiplex_barcodes/demultiplex_barcodes/64d1bebf-3811-4c21-bc1e-cf0c43f65fe8/call-barcode_report/execution/barcode_summary.csv"
  },
  "workflowRoot": "/pbi/dept/secondary/siv/smrtlink/smrtlink-bihourly/jobs-root/cromwell-executions/pb_demux_subreads_auto/4e85a878-6316-40f5-82e4-b44c46691768",
  "id": "4e85a878-6316-40f5-82e4-b44c46691768",
  "submission": "2019-06-01T15:19:36.004Z",
  "status": "Succeeded"
}
```

Job Directory:

Directory structure is significantly different because the directory numbering convention on the SMRT Link side was changed. Also, the Cromwell workflow output is more complex than `pbsmrtpipe's`, and may have arbitrary levels of nesting depending on how the workflow was written.

The Cromwell execution happens in a different directory, which is then symlinked in SMRT Link. We recommend that you use the `datastore.json` as the primary interface for identifying output files and avoid hard-coding assumptions about file names. For user convenience, both SMRT Link and `pbcromwell` create an `outputs` subdirectory with symlinks to all of the Cromwell workflow outputs. Please do **not** make any assumptions about the structure of the actual `cromwell` execution directory, as this may change in future SMRT Link releases.

```
$ ls jobs-root/0000/0000000/0000000028
cromwell-job    entry-points  metadata-summary.json  pbscala-job.stderr
datastore.json  logs          outputs                pbscala-job.stdout
```

Reports:

Reports are unchanged, except for their datastore source IDs.

Run Design CSV:

The format has **not** changed, only the appropriate values for auto-analysis pipeline ID and associated task options. Entry points are identical to the old applications, except that a reference file is optional for `pb_isoseq3`.

REST API:

The only major change is that the job type ID becomes `analysis`, and all `pbsmrtpipe` jobs are retroactively changed to this job type.

Secondary Analysis Output Files

This is data produced by secondary analysis, which is performed on the primary analysis data generated by the instrument.

- All files for a specific analysis reside in **one** directory named according to the analysis job ID number.
- Every analysis result has the following file structure. **Example**:

```
$SMRT_ROOT/userdata/jobs_root/0000/0000000/0000000002/
├── cromwell-job -> $SMRT_ROOT/userdata/jobs-root/cromwell-executions/
    pb_demux_subreads_auto/24e691c8-8d0d-4670-9db3-c7cb1126e8f8
├── entry-points
│   └── ae6f1c2c-b4a2-41cc-8e44-98b494f12a57.subreadset.xml
├── logs
│   ├── pb_simple_mapping
│   │   └── 24e691c8-8d0d-4670-9db3-c7cb1126e8f8
│   │       ├── call-mapping
│   │       │   └── execution
│   │       │       ├── stderr
│   │       │       └── stdout
│   └── workflow.24e691c8-8d0d-4670-9db3-c7cb1126e8f8.log
├── outputs
│   ├── mapping.report.json -> $SMRT_ROOT/userdata/jobs-root/cromwell-executions/
        pb_simple_mapping/24e691c8-8d0d-4670-9db3-c7cb1126e8f8/call-mapping/execution/
        mapping.report.json
│   └── mapped.bam -> $SMRT_ROOT/userdata/jobs-root/cromwell-executions/
        pb_simple_mapping/24e691c8-8d0d-4670-9db3-c7cb1126e8f8/call-mapping/execution/
        mapped.bam
├── pbscala-job.stderr
├── pbscala-job.stdout
└── workflow
    ├── analysis-options.json
    ├── datastore.json
    ├── engine-options.json
    ├── inputs.json
    ├── metadata.json
    ├── metadata-summary.json
    ├── task-timings.metadata.json
    └── timing-diagram.html
```

- `logs/`: Contains log files for the analysis job.
  - `workflow.<UUID>.log`: Global log of each significant step in the analysis and snippets from a task's `stderr` output if the analysis failed.
  - The same directory contains `stdout` and `stderr` for individual tasks.
- `cromwell-job/`: Symbolic link to the actual Cromwell execution directory, which resides in another part of the `jobs-root` directory. Contains subdirectories for each workflow task, along with executable scripts, output files, and `stderr`/`stdout` for the task.
  - `call-tool_name/execution/`: Example of an individual task directory (This is replaced with `<task_id>` below.)
  - `<task_id>/stdout`: General task `stdout` log collection.
  - `<task_id>/stderr`: General task `stderr` log collection.
  - `<task_id>/script`: The SMRT Tools command for the given analysis task.
  - `<task_id>/script.submit`: The JMS submission script wrapping `run.sh`.
  - `<task_id>stdout.submit`: The `stdout` collection for the `script.submit` script.
  - `<task_id>/stderr.submit`: The `stderr` collection for the `script.submit` script.
- `workflow/`: Contains JSON files for analysis settings and workflow diagrams.
  - `datastore.json`: JSON file representing all output files imported by SMRT Link.

- `outputs/`: A directory containing symbolic links to all datastore files, which residue in the Cromwell execution directory. This is provided as a convenience and is **not** intended as a stable API; note that external resources from dataset XML and report JSON file are **not** included here.
- `pbscala-job.stderr`: Log collection of `stderr` output from `pbscala`.
- `pbscala-job.stdout`: Log collection of `stdout` output from `pbscala`. (**Note**: This is the file displayed as **Data > SMRT Link Log** on the Analysis Results page.)

P/N 101-850-100 version 02 (October 2019)