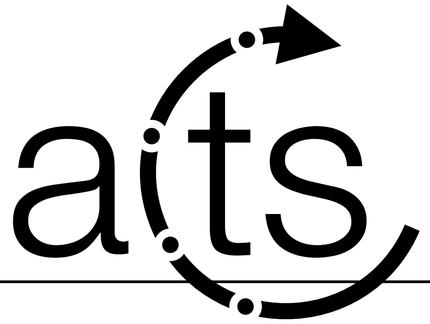
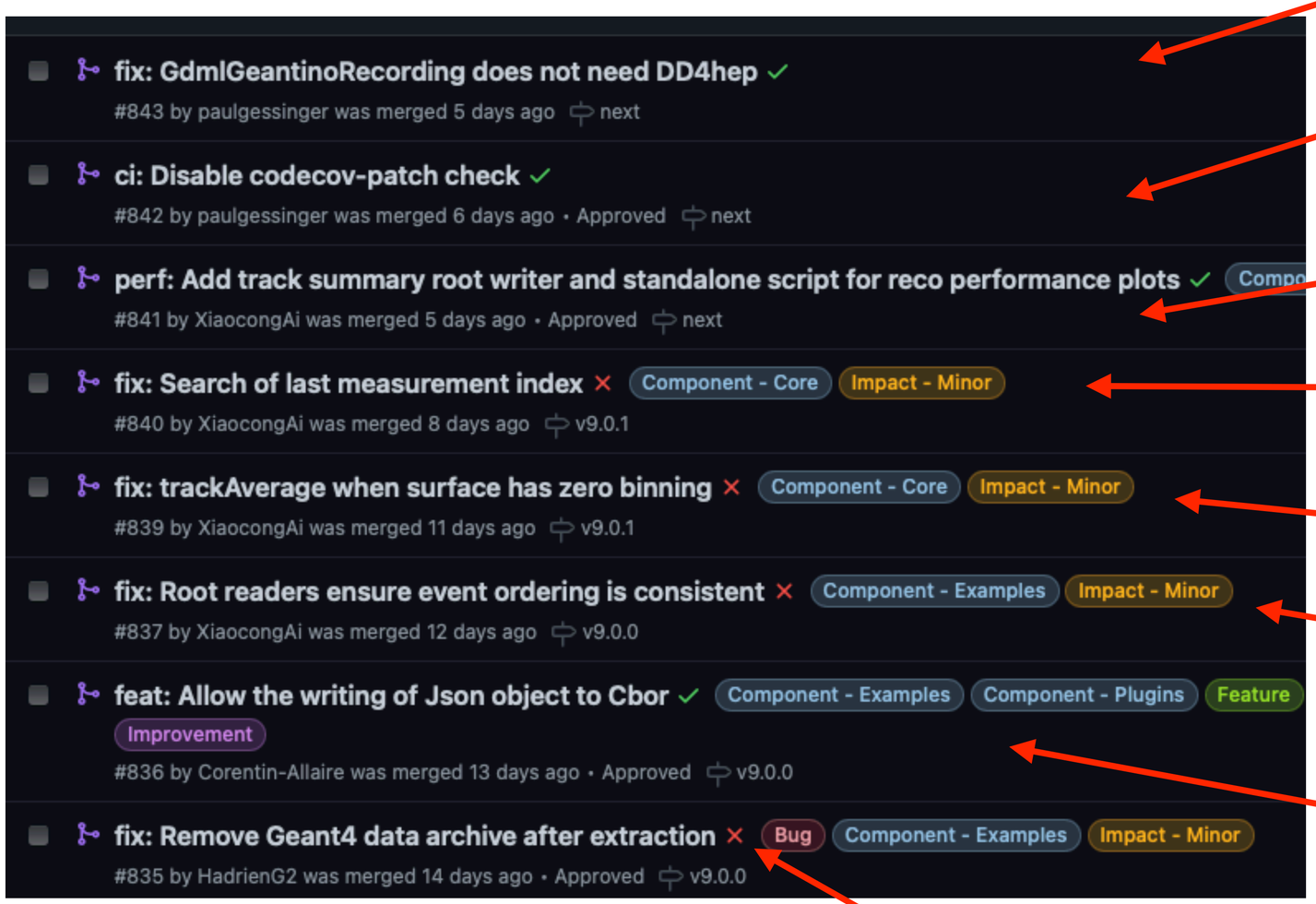


Introduction & News

A. Salzburger (CERN)

PRs | 08/06/21 - 22/06/21



Decouples Gdml building from DD4hep

Codecov-check is switched off

Performance writing update

Fix to CKF

Bugfix to CKF

Work on ROOT reading for performance tuning

Possibility to write binary JSON format

Cleanup of G4 data folder

Release v9.0.0 | 10/06/21

⚠️ BREAKING CHANGES

- Refactor of the template for the input measurements in the CKF `findTracks` invocation. (b00d3dd, #830)
  It implements a source link accessor concept which has:
  - type members including `Container`, `Key`, `Value`, `Iterator`
  - lookup methods including `count`, `range` and `at` to help access the source link container

  The CKF `findTracks` is changed to be templated on the source link accessor. In this way, the input measurements can be directly used by the CKF Actor without further internal converting as long as proper accessor is provided.

- The constructor and public members of `Acts::Navigator` change (641e00c, #826)
  When before it could be created like

```
Acts::Navigator nav{tGeometry};

nav.resolveSensitive = true;

nav.resolveMaterial = true;

nav.resolvePassive = true;
```

  it must now be created like

```
Acts::Navigator::Config cfg;

cfg.trackingGeometry = tGeometry;

nav.resolveSensitive = true;

nav.resolveMaterial = true;

nav.resolvePassive = true;

Acts::Navigator nav{cfg};
```

  Since `trackingGeometry` is the first member of `Acts::Navigator::Config`, if you don't want to change the `resolve*` values, you can also write

```
Acts::Navigator nav{{tGeometry}};
```

Measurement provider for CKF becomes template

Constructor of navigator changed

B Field access & instantiation changed

- The instantiation of an interpolated B field changes. (ba385e3, #828)
  Old:

```
using Grid_t = detail::Grid<Vector3, detail::EquidistantAxis, detail::EquidistantAxis>;

using Mapper_t = InterpolatedBFieldMapper<Grid_t>;

using BField_t = InterpolatedBFieldMap<Mapper_t>;


Grid_t g(std::make_tuple(std::move(r), std::move(z)));

Mapper_t mapper(transformPos, transformBField, std::move(g));

BField_t::Config config(std::move(mapper));

config.scale = 1.;


BField_t b(std::move(config));
```

  New:

```
using Grid_t = detail::Grid<Vector3, detail::EquidistantAxis, detail::EquidistantAxis>;

using BField_t = InterpolatedBFieldMap<Grid_t>;


Grid_t g(std::make_tuple(std::move(r), std::move(z)));

BField_t::Config cfg;

cfg.transformPos = transformPos;

cfg.transformBField = transformBField;

cfg.grid = std::move(g);

cfg.scale = 1.;
```

- B field access returns Result (b6371e2, #825)
  - The signature of field query methods in `MagneticFieldProvider` changes from

```
virtual Vector3 getField(const Vector3& position, Cache& cache) const = 0;

virtual Vector3 getFieldGradient(const Vector3& position, ActsMatrix<3, 3>& derivative, Cache& cache) const = 0;
```

    to

```
virtual Result<Vector3> getField(const Vector3& position, Cache& cache) const = 0;

virtual Result<Vector3> getFieldGradient(const Vector3& position, ActsMatrix<3, 3>& derivative, Cache& cache) const = 0;
```

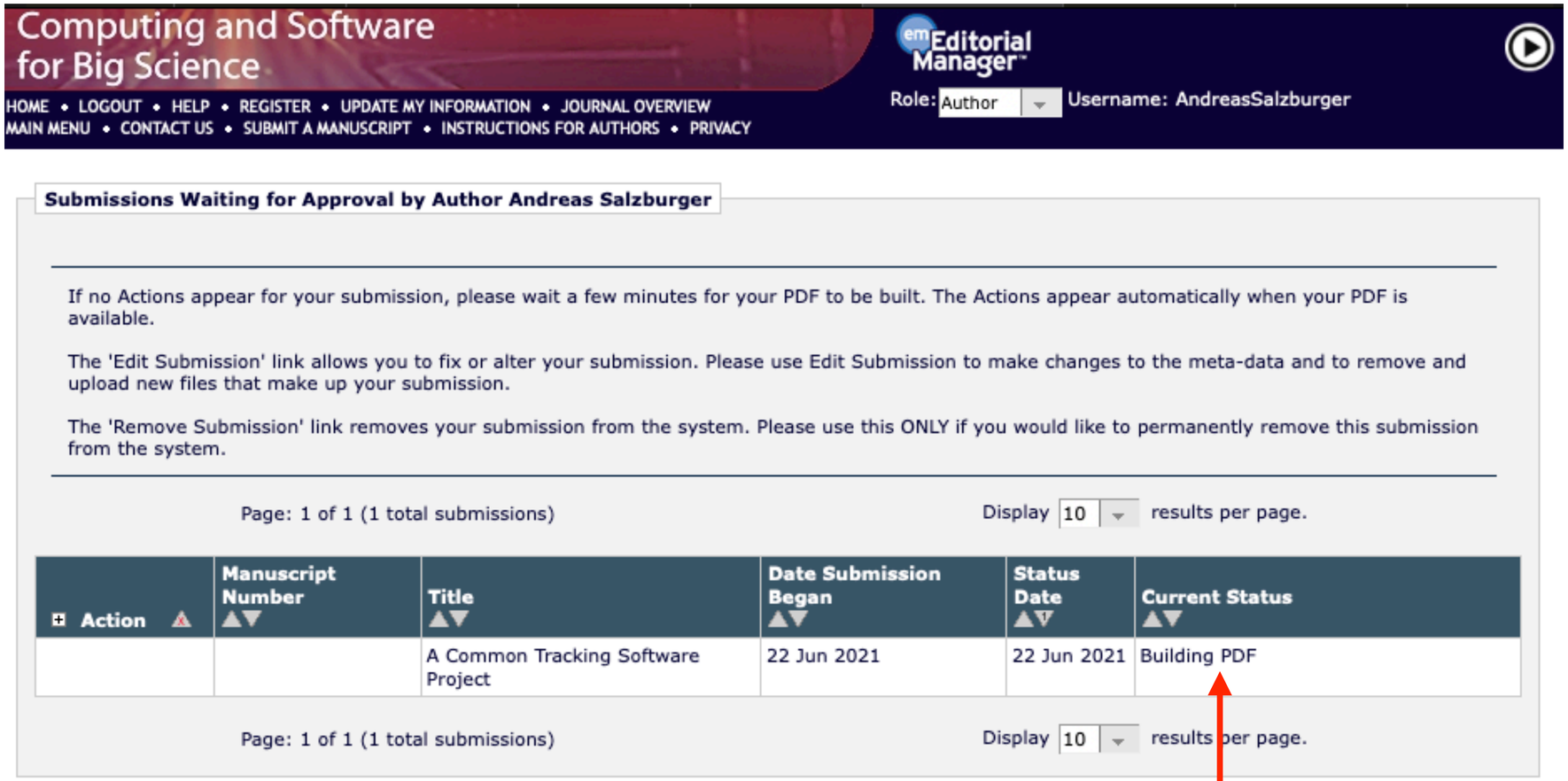
  - `InterpolatedBFieldMap::getMin` and `InterpolatedBFieldMap::getMax` now return the extent of the valid **interpolation domain**, rather than the raw grid extent.
- `Acts::MagneticFieldProvider` loses two pure virtual overloads (815bb72, #819)

```
virtual Vector3 getField(const Vector3& position) const = 0;

virtual Vector3 getFieldGradient(const Vector3& position, ActsMatrix<3, 3>& derivative) const = 0;
```

  Clients of generic magnetic field providers need to be adapted.

Finally ...



… on the brink of submission!

Upcoming meetings | 22/06/21 - 29/06/21