

BmnRoot DST Format

Reference guide to the reconstructed data
of the BM@N experiment for physics analysis

Document	Description of the DST (Data Summary Tape) format, version 1.0
Date	28 September 2026
Experiment	BM@N (Baryonic Matter at Nuclotron), NICA facility, JINR
Software	BmnRoot, dev branch (commit 945246e4)
Applies to	Simulated and experimental DSTs (one common format)

Contents

1 Introduction.....	3
1.1 Purpose and scope	3
1.2 Two processing chains, one DST format	3
1.3 Reconstruction tasks and their DST output.....	3
1.4 Conventions	4
2 File structure	5
2.1 Objects stored in the file.....	5
2.2 The event tree bmndata	5
2.3 Links between objects.....	6
3 Run-level information	7
3.1 DstRunHeader	7
3.2 Geometry and parameter containers	7
4 Event header, trigger and centrality	8
4.1 DstEventHeader.....	8
4.2 BmnTrigInfo.....	9
4.3 EventCentrality	10
5 Tracks and vertices.....	10
5.1 Track parameters (FairTrackParam)	10
5.2 Global tracks: BmnGlobalTrack	10
5.3 Inner tracks: StsVector	12
5.4 Primary vertices	12
5.5 Beam tracks: BmnBeamTrack	13
6 Hits and clusters.....	13
6.1 Common hit fields (FairHit, BmnHit)	14
6.2 Hits of the strip detectors	14
6.3 Strip clusters (StripCluster)	15
6.4 Tracking hits: StsHit.....	15
6.5 TOF-400 and TOF-700 hits.....	16
7 Forward detectors.....	17
7.1 FHCAL: FHCALEvent.....	17
7.2 Scintillation Wall: ScWallEvent.....	17
7.3 Forward hodoscope: HodoEvent.....	17
7.4 Neutron detector: NdetEvent	17
8 Monte Carlo information (simulated DSTs)	18
8.1 BmnMCInfo. and MC tracks	18
8.2 Matching of reconstructed tracks to MC tracks	18
8.3 NdetSurfacePoint	19
9 Physics analysis of reconstructed data	19
9.1 Environment	19
9.2 Recommended: analysis tasks in run_ana_bmn.C.....	19
9.3 Alternative: reading a DST file directly.....	21
9.4 Quick look with TTree::Draw.....	22
9.5 Reading only the needed branches.....	22
9.6 Further examples	22
Appendix A. Summary of the DST content	23
Appendix B. Detector identifiers (DetectorId)	26
Appendix C. Differences between simulated and experimental DSTs	27
Appendix D. Fields that are not filled or need care.....	27
Appendix E. Data volume per event	28
E.1 Simulated data	28
E.2 Experimental data	29

1 Introduction

1.1 Purpose and scope

The DST (Data Summary Tape) is the main output of the BmnRoot reconstruction and the input of physics analyses in the BM@N experiment. This document describes the DST format: the layout of the ROOT file, every branch of the event tree, the stored data classes and their fields, the links between the stored objects, and the differences between DSTs of simulated and experimental events.

The DST format is **common for simulated and experimental data**: both are produced by the same reconstruction macro, written into the same tree and stored with the same classes. A few branches exist in only one type of DST: Monte Carlo information is stored only for simulated events, and beam tracks only for experimental ones. Some fields are also filled differently. Appendix A gives a short overview of all objects of a DST file with their main methods, Appendix C summarises all differences, and Appendix D lists the fields that are currently not filled or need care.

The description corresponds to the BmnRoot [dev](#) branch as of September 2026.

1.2 Two processing chains, one DST format

The DST is the end point of two processing chains that share the reconstruction step:

Stage	Simulated data	Experimental data
Input	Output of an event generator: the particles produced in the collision with their momenta	Raw data files written by the BM@N data acquisition system
First step	<code>run_sim_bmn.C</code> : macro simulates passing of the generated particles through the BM@N setup with Geant4 transport package, producing Monte Carlo tracks, their points and detector responses (digits)	<code>run_raw_bmn.C</code> macro converts the raw data to the ROOT format and decodes to digits
Intermediate file	Simulation file: MC tracks, MC points, digits	Digit file
Reconstruction	<code>run_reco_bmn.C</code> macro reconstructs event data: hits, tracks, vertices, particle identification information, data of the forward detectors, centrality information	<code>run_reco_bmn.C</code> – the same macro with the similar output format
Output	DST file with the reconstructed data of the simulated events	DST file with the reconstructed data of the experimental events

`run_reco_bmn.C` recognises the type of its input file automatically (`BmnFunctionSet::isSimulationFile`). For experimental data the geometry is loaded from the Geometry Database, and the magnetic field scale, beam, target and run times are taken from the Condition Database (UniConDa). For simulated data this information comes from the metadata of the input simulation file.

1.3 Reconstruction tasks and their DST output

Every DST branch is produced by a task of `run_reco_bmn.C`. The table lists the tasks for run periods ≥ 8 in the order in which they are executed in `run_reco_bmn.C`. Availability: **Both** – simulated and experimental DSTs, **Sim** – simulated only, **Exp** – experimental only.

Task (class)	DST branch(es)	Content	Avail.
<code>BmnFillTrigInfoDst</code>	<code>BmnTrigInfo</code>	Trigger and beam counter information	Both
<code>BmnSiBTHitMaker</code>	<code>BmnSiBTHit</code>	Hits in the Silicon Beam Tracker upstream of the target	Exp
<code>BmnVSPHitMaker</code>	<code>BmnVSPHit</code> , <code>BmnVSPUpper/LowerCluster</code>	Vertex Silicon Plane hits and clusters	Both
<code>BmnSiliconHitMaker</code>	<code>BmnSiliconHit</code> , <code>BmnSiliconUpper/LowerCluster</code>	Forward Silicon Detector hits and clusters	Both
<code>BmnGemStripHitMaker</code>	<code>BmnGemStripHit</code> , <code>BmnGemUpper/LowerCluster</code>	GEM hits and clusters	Both

Task (class)	DST branch(es)	Content	Avail.
BmnCSCHitMaker	BmnCSCHit, BmnCSCUpper/LowerCluster	Cathode Strip Chamber hits and clusters	Both
BmnTof1HitProducer	BmnTof400Hit, BmnTof700Hit	TOF-400 and TOF-700 hits	Both
BmnFHCALReconstructor	FHCALEvent	Forward Hadron Calorimeter	Both
BmnScWallReconstructor	ScWallEvent	Scintillation Wall	Both
BmnHodoReconstructor	HodoEvent	Forward (fragment) hodoscope	Both
BmnNdetReconstructor	NdetEvent; NdetSurfacePoint (Sim)	Neutron detector	Both
BmnToCbmHitConverter	StsHit	VSP, Silicon and GEM hits prepared for tracking	Both
BmnStsVectorFinderV9, BmnStsMatchTracks	StsVector; StsTrackMatch (Sim)	Inner tracks (VSP + Silicon + GEM) and their MC matching	Both
BmnBeamTracking	BmnBeamTrack	Beam ion tracks in the SiBT	Exp
CbmFindPrimaryVertex (CbmPVFinderKF)	PrimaryVertex.	Primary vertex, CBM KF finder	Both
BmnKFPrimaryVertexFinder	MpdVertex.	Primary vertex, default for global tracks	Both
BmnCombineVertexFinder	CombineVertex	Vertices combined with the beam track	Exp
BmnGlobalTracking	BmnGlobalTrack	Global tracks with TOF matching	Both
BmnMatchRecoToMC	BmnGlobalTrackMatch	Matching of global tracks to MC tracks	Sim
BmnDedx	fields of BmnGlobalTrack	dE/dx and charge estimate (period 8 only)	Both
CentralityClusterizer	EventCentrality	Centrality class	Both
BmnFillDSTTask	DstEventHeader., DstRunHeader; BmnMCInfo. (Sim)	Event and run headers, MC event information and MC tracks	Both

Note: If the reconstruction is run with the `L1Tracking` option instead of the default `VectorFinder`, the inner tracks are stored in the branch `StsTrack` instead of `StsVector` (same class `CbmStsTrack`). For experimental data of period 8 the pile-up classifier `BmnEventSelector` also runs and can add the branch `BmnBC1hitInfo..` Run periods ≤ 7 use a different chain (MWPC, DCH, ZDC) that is not covered here.

1.4 Conventions

- **Units:** coordinates and lengths in cm, momenta in GeV/c, energies in GeV, times in ns, angles in rad, unless stated otherwise.
- **Coordinate system:** global BM@N (laboratory) system with the z axis along the beam direction; the nominal target position is near $z = 0$.
- **Events:** one entry of the tree `bmndata` is one event; all objects of an event are in the same entry.
- **Index links:** integer links such as `GetTof1HitIndex()` are positions (starting from 0) in the `TClonesArray` of another branch of the same event. The value `-1` means “no link”.
- **Branch names:** names ending with a dot (for example `DstEventHeader.`) are single objects; names without a dot are `TClonesArray` collections, except `EventCentrality`, `FHCALEvent`, `HodoEvent`, `NdetEvent` and `ScWallEvent`, which are single objects too.
- **Getters and members:** the tables give the getter to be used and, where useful, the data member name (prefix `f`). The branches are split, so member names can be used directly in `TTree::Draw`, for example `bmndata->Draw("BmnGlobalTrack.fParamFirst.fQp")`.
- **Availability** column: **Both**, **Sim** (simulated DST only) or **Exp** (experimental DST only). Remarks such as “Sim: ...; Exp: ...” describe fields that exist in both but are filled differently.

2 File structure

2.1 Objects stored in the file

Besides the event tree, a DST file contains run-level objects written once per file:

Key	Class	Content	Avail.
bmndata	TTree	Event tree: one entry per reconstructed event (Section 2.2)	Both
DstRunHeader	DstRunHeader	Run information: period, run, beam, target, energy, field (Section 3.1)	Both
FairGeoParSet	FairGeoParSet	Full geometry (TGeoManager) used in the reconstruction; the description holds the geometry tag	Both
run ID	FairRtdbRun	Runtime database bookkeeping; the key name is the run ID (period number for MC)	Both
FileHeader	FairFileHeader	FairRoot file header	Both
BranchList, TimeBasedBranchList	TList	FairRoot bookkeeping: names of all branches known during the reconstruction, including input branches that are not stored in the DST	Both
bmnroot	TFolder	FairRoot folder structure: which branch belongs to which detector folder	Both
FairBaseParSet	FairBaseParSet	Simulation run parameters: beam energy, description "beam;target"	Sim
BmnFieldPar	BmnFieldPar	Magnetic field parameters of the simulation	Sim
FairGeoPassivePar	FairGeoPassivePar	Passive geometry parameters of the simulation	Sim

Note: A file can contain several cycles of the tree key (for example `bmndata;2` and `bmndata;3`). ROOT automatically uses the latest cycle when the tree is read by name.

2.2 The event tree bmndata

The table lists all branches of `bmndata` with the class of the object (for `TClonesArray` branches, the class of the elements), the FairRoot folder and the availability.

Branch	Class	Folder	Content	Avail.
DstEventHeader.	DstEventHeader	EvtHeader	Event header	Both
BmnTrigInfo.	BmnTrigInfoDst	TRIGGER	Trigger, beam counters, T0	Both
BmnVSPHit	BmnVSPHit	VSP	Vertex Silicon Plane hits	Both
BmnVSPUpperCluster, BmnVSPLowerCluster	StripCluster	VSP	VSP strip clusters	Both
BmnSiliconHit	BmnSiliconHit	SILICON	Forward Silicon Detector hits	Both
BmnSiliconUpperCluster, BmnSiliconLowerCluster	StripCluster	SILICON	Silicon strip clusters	Both
BmnGemStripHit	BmnGemStripHit	GEM	GEM hits	Both
BmnGemUpperCluster, BmnGemLowerCluster	StripCluster	GEM	GEM strip clusters	Both
BmnCSCHit	BmnCSCHit	CSC	Cathode Strip Chamber hits	Both
BmnCSCUpperCluster, BmnCSCLowerCluster	StripCluster	CSC	CSC strip clusters	Both
BmnTof400Hit	BmnTofHit	TOF400	TOF-400 hits	Both
BmnTof700Hit	BmnTofHit	TOF700	TOF-700 hits	Both
StsHit	CbmStsHit	STSHIT	Tracking hits (VSP + Silicon + GEM)	Both
StsVector	CbmStsTrack	STS	Inner tracks	Both
PrimaryVertex.	CbmVertex	Global	Primary vertex (CBM KF finder)	Both

Branch	Class	Folder	Content	Avail.
MpdVertex.	CbmVertex	Global	Primary vertex (default for global tracks)	Both
BmnGlobalTrack	BmnGlobalTrack	GLOBAL	Global tracks	Both
EventCentrality	BmnEventCentrality	–	Centrality class	Both
FHCalEvent	BmnFHCalEvent	–	FHCal modules	Both
HodoEvent	BmnHodoEvent	–	Hodoscope: strips	Both
NdetEvent	BmnNdetEvent	–	Neutron detector cells	Both
ScWallEvent	BmnScWallEvent	–	ScWall: 174 cells	Both
BmnSiBTHit	BmnSiBTHit	SiBT	Silicon Beam Tracker hits	Exp
BmnBeamTrack	BmnTrack	BEAM	Beam tracks (SiBT)	Exp
CombineVertex	CbmVertex	VERTEX	Vertices combined with the beam track	Exp
NdetSurfacePoint	BmnNdetSurfacePoint	NDET	MC points on the neutron detector surfaces	Sim
StsTrackMatch	CbmTrackMatch	STS	MC matching of inner tracks	Sim
BmnGlobalTrackMatch	BmnTrackMatch	GLOBAL	MC matching of global tracks	Sim
BmnMCInfo.	BmnMCInfoDst	MC	MC event information and MC tracks	Sim

Tip: Hits and clusters make up most of the data volume of a DST (Appendix E). Analyses that need only tracks, vertices and forward detectors read much faster if the hit and cluster branches are not read (Section 9.5).

2.3 Links between objects

The objects of an event are connected through integer indices. The most important links are:

From	Accessor	To	Avail.
BmnGlobalTrack	GetGemTrackIndex()	StsVector: the inner track of the global track	Both
BmnGlobalTrack	GetTof1HitIndex(), GetTof2HitIndex()	BmnTof400Hit, BmnTof700Hit	Both
BmnGlobalTrack	GetScWallCellId()	ScWallEvent cell (1–174), GetCell(id)	Both
StsVector (CbmStsTrack)	GetStsHitIndex(i), i < GetNStsHits()	StsHit	Both
StsHit (CbmStsHit)	fDigiF % 1000000, fDigiB % 1000000	Lower and upper cluster of the hit in Bmn<Det>LowerCluster / Bmn<Det>UpperCluster, detector given by GetSystemId() (Section 6.4)	Both
Strip hits (BmnHit)	GetUpperClusterIndex(), GetLowerClusterIndex()	Bmn<Det>UpperCluster, Bmn<Det>LowerCluster of the same detector	Both
MpdVertex.	GetTrackInds() (member trkID)	StsVector: tracks used in the vertex fit	Both
StsHit	–GetRefIndex() – 1	Original hit in BmnVSPHit / BmnSiliconHit / BmnGemStripHit (by GetSystemId())	Exp
BmnBeamTrack	GetHitIndex(i), i < GetNHits()	BmnSiBTHit	Exp
BmnGlobalTrack	GetRefIndex()	BmnMCInfo. MC track array	Sim
BmnGlobalTrackMatch[i]]	same index i	BmnGlobalTrack[i]	Sim
StsTrackMatch[i]	same index i	StsVector[i]	Sim
StsVector	GetTrkID()	BmnMCInfo. MC track array	Sim
CbmMCTrack	GetMotherId()	Mother MC track in the same array (–1 for primary particles)	Sim

3 Run-level information

3.1 DstRunHeader

`DstRunHeader` is written once per file by `BmnFillDSTTask`. For experimental data it is filled from the Condition Database, for simulated data from the metadata of the input simulation file.

Getter (member)	Type	Description	Sim / Exp
<code>GetPeriodId()</code> (<code>fPeriodNumber</code>)	UInt_t	Run period (session) number, e.g. 9	Both
<code>GetRunId()</code> (<code>fRunNumber</code>)	UInt_t	Run number	Both; Sim: 0
<code>GetBeam()</code>	TString	Beam particle, e.g. "Xe"; empty string – not defined	Both
<code>GetBeamA()</code> , <code>GetBeamZ()</code>	Int_t	Mass number and charge of the beam ion (–1 – unknown)	Both
<code>GetBeamEnergy()</code>	Float_t	Beam energy per nucleon [GeV], e.g. 3.0	Both; Sim: from the generator file name
<code>GetTarget()</code>	TString	Target, e.g. "Sn", "CsI (2%)", "Empty"; empty string – not defined	Both
<code>GetTargetA()</code> , <code>GetTargetZ()</code>	Int_t	Target A and Z. A = 0 – empty target; Z = 0 – compound target (A is then the sum of the atomic weights, e.g. 260 for CsI) or empty target; –1 – not defined	Both
<code>GetFieldType()</code>	Int_t	Magnetic field type: 0 – constant field, 1 – field map	Both
<code>GetFieldScale()</code>	Double_t	Scale factor of the field map (–1 for a constant field). Exp: magnet voltage from the database divided by the map voltage, with an energy-dependent correction	Both
<code>GetFieldName()</code>	TString	Field map file; for a constant field the string "xmin:xmax:ymin:ymax:zmin:zmax:Bx:By:Bz"	Both
<code>GetStartTime()</code> , <code>GetFinishTime()</code>	TTimeStamp	Start and end of the run from the Condition Database. The database stores local (Moscow) time; the values are put into <code>TTimeStamp</code> as if they were UTC	Exp; Sim: time of the DST production, not meaningful

3.2 Geometry and parameter containers

The geometry used in the reconstruction is stored in the container `FairGeoParSet`. Its description holds the geometry tag of the Geometry Database (e.g. `run9_26.09.0`). When the DST is processed with `FairRunAna` (for example, in `run_ana_bmn.C`), the geometry is loaded automatically as `gGeoManager` during `fRunAna->Init()`, so the detector volumes and materials are available in the analysis tasks, e.g. for track extrapolation. Only if `FairRunAna` is not used, for example when the DST file is read directly, the geometry can be taken from the container itself, which is seldom needed:

```
TFile* file = TFile::Open("bmndst.root");
FairGeoParSet* geoPar = (FairGeoParSet*)file->Get("FairGeoParSet");
cout << "Geometry tag: " << geoPar->getDescription() << endl;
TGeoManager* geometry = geoPar->GetGeometry(); // the same object as gGeoManager
```

Simulated DSTs also contain `FairBaseParSet`, whose beam momentum field holds the beam energy per nucleon and whose description holds "beam;target" (e.g. "Xe;Sn"). This is where `DstRunHeader` takes the beam and target of simulated data from.

4 Event header, trigger and centrality

4.1 DstEventHeader.

The event header (`DstEventHeader`, derived from `FairEventHeader`) is filled by `BmnFillDSTTask`: from the MC event header for simulated data and from the digit event header (`BmnEventHeader`) for experimental data.

Getter (member)	Type	Description	Sim / Exp
<code>GetRunId()</code> (<code>fRunId</code>)	UInt_t	Exp: run number. Sim: run ID of the simulation	Both
<code>GetEventId()</code> (<code>fEventId</code>)	UInt_t	Exp: event number from the DAQ; unique within a run, not consecutive in the DST. Sim: MC event number, starting from 1	Both
<code>GetPeriodId()</code> (<code>fPeriodId</code>)	UInt_t	Run period. Exp: from the digit event header; Sim: from the simulation file	Both
<code>GetEventTime()</code> (<code>fEventTime</code>)	Double_t	Exp: absolute (astronomical) time of the event in UTC as UNIX time: seconds since 1970-01-01, equal to <code>GetEventTimeTS().AsDouble()</code> . At present dates the double value resolves about 0.24 μ s. Sim: MC event time of the generator, not an absolute time (0 if the generator does not set it)	Both
<code>GetEventTimeTS()</code> (<code>fEventTimeTS</code>)	TTimeStamp	Exp: the same moment as <code>GetEventTime()</code> as a time stamp in UTC with nanosecond precision: <code>GetSec()</code> , <code>GetNanoSec()</code> . Sim: the MC event time converted to a time stamp (1970-01-01 00:00:00 for time 0)	Both
<code>GetTriggerType()</code> (<code>fTriggerType</code>)	BmnTriggerType	Trigger type: <code>kBMNBEAM</code> = 6, <code>kBMNMINBIAS</code> = 1. Currently not filled : always 1. Use the trigger input signals of <code>BmnTrigInfo</code> . for trigger selection	Both (not filled)
<code>GetZ2in()</code> , <code>GetZ2out()</code> , <code>GetADCin()</code> , <code>GetADCout()</code> , <code>GetZin()</code>	Double_t, Short_t	Beam ion charge squared (Z^2) and ADC signals before (BC1, BC2) and after (BC3, BC4) the target. Computed only when the T0 and BC1–BC4 TQDC digit branches exist in the input; otherwise –100	Exp (older runs)
<code>GetADC1()</code> ... <code>GetADC4()</code>	Double_t	Single beam counter amplitudes for 2D charge cuts; same conditions as above, –100 otherwise	Exp (older runs)
<code>GetB()</code> (<code>fB</code>)	Double_t	Impact parameter [fm] from the generator	Sim; Exp: 0
<code>GetMCEntryNumber()</code> , <code>GetInputFileId()</code>	Int_t	FairRoot bookkeeping: entry number of the source event in the simulation file; input file index	Sim; Exp: 0
<code>GetZ1()</code> ... <code>GetZ4()</code>	Double_t	Reserved (charge from single counters), not used	–

4.2 BmnTrigInfo.

BmnTrigInfoDst collects trigger-related information of the event: trigger input signals, amplitudes of the beam counters and of the fragment detector, the T0 start time, and the corresponding digits. In simulated DSTs only the Barrel Detector digits and the Fragment Detector MC points are available.

Getter (member)	Type	Description	Sim / Exp
GetBDMult(), GetSiMDMult()	UInt_t	Number of fired channels of the Barrel Detector (BD) and of the Silicon Multiplicity Detector (SiMD); 0 if the corresponding digit branch is absent in the input	Both; SiMD: Exp
GetFDDigits()	TClonesArray*	Exp: BmnTrigWaveDigit of the FD. Sim: BmnFDPoint (MC points in the FD)	Both
GetBDDigits()	TClonesArray* of BmnTrigDigit	Barrel Detector digits	Both
GetInputSignalsAR(), IsTriggerBitTrueAR(bit)	UInt_t	Bit mask of the trigger input signals after reduction (prescaling)	Exp; Sim: 0
GetInputSignalsBR(), IsTriggerBitTrueBR(bit)	UInt_t	Bit mask of the trigger input signals before reduction	Exp; Sim: 0
GetInputSignalsVector()	vector<uint32_t>	Trigger input signal words as delivered by the trigger system	Exp; Sim: empty
GetT0_time(), GetT0_amp()	Float_t	Start time T0 [ns, TDC time scale] and its amplitude, reconstructed from the BC1 top and bottom waveforms with slewing correction; the start time of the TOF measurement. Available since Run 9	Exp (Run 9+); Sim: 0
GetVCamp()	Float_t	Peak amplitude of the veto counter (VC)	Exp
GetBC0Amp(), GetBC1Amp(), GetBC2Amp(), GetBC1Integral()	Float_t	Peak amplitudes of the beam counters BC0, BC1, BC2 and the integral of the BC1 signal [ADC counts]	Exp
GetFDAmp()	Float_t	Peak amplitude of the Fragment Detector (FD)	Exp
GetBC0Digits(), GetBC1Digits(), GetBC2Digits()	TClonesArray* of BmnTrigWaveDigit	Waveform digits of the beam counters	Exp

Classes of the trigger digits:

Class	Fields
BmnTrigWaveDigit	fMod channel; fNsmpl number of samples; fValue waveform samples (GetValue()); fTime time; fTdcTimes TDC times (TdcVector()); fTrigTimestamp, fAdcTimestamp time stamps; GetPeak(), GetIntegral()
BmnTrigDigit	fMod channel (40 channels for BD); fTime time [ns]; fAmp amplitude; fTimestamp
BmnFDPoint	FairMCPoint (track index, position, momentum, time, energy loss) plus exit position and momentum, fCharge, fPdgId, fIsPrimary, fLightYield, fTimeIn, fTimeOut, fLengthtrack

4.3 EventCentrality

`BmnEventCentrality` holds the centrality classification of the event made by `CentralityClusterizer`. The event is classified in the plane (FHCAL total energy, hodoscope total signal) using two-dimensional probability density functions of 7 clusters. Every cluster gets a probability, and the class is the most probable cluster. If the event lies outside all distributions, the class is the cluster with the nearest centroid, and the probability is set to 0.

Getter	Type	Description
<code>GetClass()</code>	int	Index (0–6) of the most probable cluster, method FHCALHodo
<code>GetProbability()</code>	float	Probability of that cluster (0 if assigned by the nearest centroid)
<code>GetSoftCentrality()</code>	vector<float>	Probabilities of all clusters (7 values)
<code>GetClass(method), GetProbability(method), GetSoftCentrality(method)</code>		The same for a given method; <code>BmnCentralityClass::Method::FHCALHodo</code> is the only method

Caution: The cluster index is a label of the calibration, not an ordered centrality percentile. The classes are meaningful only if the calibration file set in `run_reco_bmn.C` was obtained for the analysed collision system and beam energy.

5 Tracks and vertices

5.1 Track parameters (FairTrackParam)

All track classes store the track state as `FairTrackParam` objects at two points: `GetParamFirst()` and `GetParamLast()`.

Getter (member)	Description
<code>GetX(), GetY(), GetZ()</code>	Position of the track state [cm]
<code>GetTx(), GetTy()</code>	Slopes $tx = dx/dz = px/pz$, $ty = dy/dz = py/pz$
<code>GetQp() (fQp)</code>	Charge over momentum q/p [(GeV/c) ⁻¹]; the sign is the charge sign
<code>GetCovariance(i, j), CovMatrix(...) (fCovMatrix[15])</code>	5 × 5 covariance matrix of (x, y, tx, ty, q/p); the 15 elements of the upper triangle are stored row by row: a[0,0..4], a[1,1..4], a[2,2..4], a[3,3..4], a[4,4]

Momentum components follow from the track parameters:

$$\begin{aligned}
 p &= 1 / |q/p| & \text{charge sign} &= \text{sign}(q/p) \\
 pz &= p / \sqrt{1 + tx^2 + ty^2} \\
 px &= tx \cdot pz, & py &= ty \cdot pz, & pt &= pz \cdot \sqrt{tx^2 + ty^2}
 \end{aligned}$$

Note: The track fit assumes unit charge: for a particle with charge $|Z| > 1$ the value $1/|q/p|$ is the rigidity $p/|Z|$, and the momentum is $|Z|/|q/p|$. The inner track fit uses the pion mass hypothesis for material effects (`CbmStsTrack::GetPidHypo() = 211`).

5.2 Global tracks: BmnGlobalTrack

Global tracks are the main objects for physics analysis. `BmnGlobalTracking` creates one global track for every inner track of `StsVector` with at least 4 hits and then matches the tracks to TOF-400, TOF-700 and ScWall hits. Global tracks are created only if the vertex `MpdVertex` was reconstructed with at least 2 tracks; otherwise the event has no global tracks.

- `GetParamFirst()` is the inner track state propagated to the z of `MpdVertex`. (the track is not refitted with the vertex constraint).
- `GetParamLast()` is the state at the last measurement: the last inner track hit or, after TOF matching, the matched TOF-400 or TOF-700 hit.

- `GetLength()` is the track length from the target plane to the last matched point (TOF-700 hit, otherwise TOF-400 hit, otherwise the last inner track hit).
- $\beta = L / (c \cdot t)$, where L is the length from the target to the TOF hit and t is the time of flight of the TOF hit; $m^2 = p^2 / (1/\beta^2 - 1)$ with $p = \text{GetP}()$. For $|Z| > 1$ this gives $(m/Z)^2$.

Getter (member)	Type	Description	Sim / Exp
<code>GetParamFirst()</code>	FairTrackParam*	Track parameters at the z of <code>MpdVertex</code> .	Both
<code>GetParamLast()</code>	FairTrackParam*	Track parameters at the last measurement (inner tracker or TOF)	Both
<code>GetP()</code>	Double_t	$1 / (q/p)$ of <code>ParamFirst</code> : signed momentum (rigidity for $ Z > 1$)	Both
<code>GetPx(), GetPy(), GetPz(), GetPt(), GetTheta(), GetPhi(), GetEta()</code>	Double_t	Kinematics computed from <code>ParamFirst</code> ; θ and ϕ in rad, η is the pseudorapidity	Both
<code>GetNHits() (fNhits)</code>	Int_t	Number of inner track hits plus one for each matched TOF hit	Both
<code>GetChi2(), GetNDF()</code>	Float_t, Int_t	χ^2 and NDF of the inner track fit	Both
<code>GetLength() (fLength)</code>	Float_t	Track length from the target to the last matched point [cm]	Both
<code>GetGemTrackIndex() (fGemTrack)</code>	Int_t	Index of the inner track in <code>StsVector</code>	Both
<code>GetTof1HitIndex() (fTof1Hit)</code>	Int_t	Index of the matched hit in <code>BmnTof400Hit</code> , -1 if none	Both
<code>GetTof2HitIndex() (fTof2Hit)</code>	Int_t	Index of the matched hit in <code>BmnTof700Hit</code> , -1 if none	Both
<code>GetBeta(1), GetBeta(2) (fBeta400, fBeta700)</code>	Double_t	β from TOF-400 (1) and TOF-700 (2); -1000 if no hit is matched. See the caution in Section 6.5 for experimental data	Both
<code>GetMass2(1), GetMass2(2)</code>	Double_t	m^2 [GeV^2/c^4] from TOF-400 / TOF-700; -1000 if no β	Both
<code>GetDCAInVertex() (fDCAInVertex)</code>	Float_t	Distance in the xy plane between <code>ParamFirst</code> and <code>MpdVertex</code> . at the vertex z [cm]	Both
<code>GetChi2InVertex() (fChi2InVertex)</code>	Float_t	χ^2 of the track–vertex compatibility (Kalman update of <code>ParamFirst</code> with <code>MpdVertex</code> . as a measurement)	Both
<code>GetScWallCellId(), GetScWallSignal()</code>	Int_t, Double_t	Matched ScWall cell (1–174) and its signal; -1 and -1000 if not matched. The matching uses a fixed ScWall position $z = 741.5$ cm (Run 8 setup)	Both
<code>GetDedx(), GetQ() (fDedx, fQ)</code>	Float_t, Short_t	dE/dx (truncated mean of calibrated cluster signals in the GEM and Silicon detectors) and charge estimate (1 or 2). Filled only for period 8 (<code>BmnDedx</code>); 0 otherwise	Both (period 8)
<code>GetRefIndex() (fRefIndex)</code>	Int_t	Sim: index of the matched MC track in <code>BmnMCInfo</code> ., -1 if less than 60 % of the hits belong to one MC track. Exp: undefined value , do not use	Sim
<code>GetVectorCscHitIndices(), GetCscHitIndex(i)</code>	vector<Int_t>	Indices of matched CSC hits in 5 zones. Currently always -1: CSC hits are not attached to tracks	–
<code>IsPrimary() (fIsPrimary)</code>	Bool_t	Not filled (always true); use <code>CbmStsTrack::FromVertex()</code>	–
<code>GetPDG() (fPDG)</code>	Int_t	Not filled (undefined value)	–
<code>GetFlag(), GetB(), GetHitIndex(i)</code>		Not filled for global tracks (-1, 0, empty hit list)	–

Tip: Primary track selection: take the inner track `StsVector[track->GetGemTrackIndex()]` and require `FromVertex()`, i.e. the track was used in the fit of `MpdVertex`.. Cuts on `GetDCAInVertex()`, `GetChi2InVertex()` and the number of hits can be added.

5.3 Inner tracks: StsVector

`StsVector` contains the tracks found by the vector finder in the inner tracker: the Vertex Silicon Plane, the Forward Silicon Detector and the GEM stations. The hits of these tracks are in `StsHit` (Section 6.4).

Getter (member)	Type	Description	Sim / Exp
<code>GetNStsHits()</code> , <code>GetStsHitIndex(i)</code> (<code>fStsHits</code>)	<code>Int_t</code>	Number of hits and their indices in <code>StsHit</code> , ordered along z	Both
<code>GetParamFirst()</code> , <code>GetParamLast()</code>	<code>FairTrackParam*</code>	Track parameters at the first and the last hit	Both
<code>GetChi2()</code> , <code>GetNDF()</code>	<code>Double_t</code> , <code>Int_t</code>	χ^2 and NDF of the Kalman filter fit	Both
<code>GetFlag()</code> , <code>FromVertex()</code>	<code>Int_t</code>	Bit 2 (value 2) is set if the track was used in the fit of <code>MpdVertex</code> .; <code>FromVertex()</code> returns <code>GetFlag()</code> & 2	Both
<code>GetPidHypo()</code>	<code>Int_t</code>	Mass hypothesis of the fit (211, pion)	Both
<code>GetB()</code> (fB)	<code>Double_t</code>	Response of the BDT (TMVA) classifier used by the track finder to reject fake tracks (set to 1 when the classifier is not applied). The header comment "impact parameter" is outdated	Both
<code>GetPt()</code> , <code>GetEta()</code>	<code>Double_t</code>	Transverse momentum and pseudorapidity from <code>ParamFirst</code>	Both
<code>GetTrkID()</code> (<code>ststrk</code>)	<code>int</code>	Sim: index of the matched MC track (-1 if not matched). Exp: undefined value	Sim
<code>fMvdHits</code> , <code>fnEv</code> , <code>fHitsArr</code>		Not used (empty)	–

5.4 Primary vertices

The DST contains several estimates of the primary vertex, all stored as `CbmVertex` objects:

Branch	Algorithm	Remarks	Avail.
<code>MpdVertex.</code>	<code>BmnKFPrimaryVertexFinder</code> : pre-estimate from histograms of extrapolated tracks, then an iterative Kalman fit with a χ^2 cut; <code>StsVector</code> tracks with ≥ 4 hits and $\chi^2 < 12.25 \cdot \text{NDF}$	Default vertex of the global tracks. The indices of the used tracks are in <code>GetTrackInds()</code> , and these tracks are flagged in <code>StsVector</code> . <code>GetUniqueID()</code> holds the number of peaks found in the z distribution of the extrapolated tracks: 0 – no clear peak, 1 – single peak, 2 – an additional peak (possible second interaction or interaction outside the target)	Both
<code>PrimaryVertex.</code>	<code>CbmFindPrimaryVertex</code> with <code>CbmPVFinderKF</code> (CBM KF vertex fit); same track selection	<code>GetTrackInds()</code> is empty. If no track is accepted, the vertex has <code>GetNTracks() = 0</code> and a covariance of 100 cm ²	Both
<code>CombineVertex</code>	<code>BmnCombineVertexFinder</code> : array of 2 vertices, [0] from <code>PrimaryVertex.</code> , [1] from <code>MpdVertex.</code> , combined with the first beam track of <code>BmnBeamTrack</code>	x and y are the weighted mean of the vertex and the beam track extrapolated to the vertex z (weights from the variances). z, χ^2 and the covariance are copied from the input vertex; <code>NTracks = input + 1</code> , <code>NDF = 2 \cdot \text{NTracks}</code> . Without a beam track the entries are copies of the input vertices	Exp

Fields of CbmVertex:

Getter (member)	Type	Description
GetX(), GetY(), GetZ(), Position(TVector3&)	Double_t	Vertex position [cm]
GetChi2(), GetNDF()	Double_t, Int_t	χ^2 and number of degrees of freedom of the vertex fit (NDF = 2·NTracks)
GetNTracks()	Int_t	Number of tracks used in the fit. Check it: 0 means that the fit failed
GetCovariance(i, j), CovMatrix(TMMatrixFSym&) (fCovMatrix[6])	Double_t	3 × 3 covariance matrix of (x, y, z); stored as a[0,0], a[0,1], a[0,2], a[1,1], a[1,2], a[2,2]
GetTrackInds() (trkID)	vector<Int_t>	Indices of the used tracks in StsVector (MpdVertex. only)

5.5 Beam tracks: BmnBeamTrack

In experimental DSTs `BmnBeamTracking` reconstructs the beam ions in the three stations of the Silicon Beam Tracker upstream of the target. Track candidates are combinations of one hit per station with signals above 200 in both strip layers; up to 10 candidates are fitted with a Kalman filter (forward and backward pass). The beam momentum is not measured: q/p is fixed.

Getter (member)	Description
GetNHits(), GetHitIndex(i)	3 hits, indices in BmnSiBTHit
GetParamFirst()	Track parameters at the first SiBT station
GetParamLast()	Track parameters propagated to z = 0 (target)
GetB()	Distance of the track from the beam axis (x = y = 0) at z = 0 [cm]
GetChi2()	Sum of the χ^2 of the forward and backward Kalman filter passes
GetQp() of the parameters	Fixed value, not a measurement
GetFlag(), GetNDF(), GetLength()	Not filled

6 Hits and clusters

The DST keeps the hits of all tracking detectors and the strip clusters, so that detector studies, calibrations and track refits can be done on the DST level. The strip detectors (SiBT, VSP, Silicon, GEM, CSC) have two strip layers (“lower” and “upper”); a hit is the crossing of a lower-layer and an upper-layer cluster.

6.1 Common hit fields (FairHit, BmnHit)

All hit classes of the tracking detectors derive from `BmnHit` (itself derived from `FairHit`).

Getter (member)	Description	Sim / Exp
<code>GetX()</code> , <code>GetY()</code> , <code>GetZ()</code>	Hit position in the global coordinate system [cm]	Both
<code>GetDx()</code> , <code>GetDy()</code> , <code>GetDz()</code>	Position errors [cm]	Both
<code>GetCovXY()</code>	Covariance of x and y [cm ²]	Both
<code>GetStation()</code> , <code>GetModule()</code>	Station and module of the hit within the detector, starting from 0	Both
<code>GetUpperClusterIndex()</code> , <code>GetLowerClusterIndex()</code>	Strip detectors: indices of the upper and lower clusters in <code>Bmn<Det>UpperCluster</code> / <code>Bmn<Det>LowerCluster</code> . TOF: see Section 6.5	Both
<code>GetRefIndex()</code>	Sim: index of the MC point that produced the hit (the MC points are not stored in the DST), -1 if there is none. Exp: -1 (TOF: see Section 6.5)	Both
<code>GetDigitNumberMatch()</code>	<code>BmnMatch</code> with two links to the central strips of the lower and upper clusters	Both
<code>IsPseudo()</code>	Pseudo hit reconstructed from a cluster in one layer only	Both
<code>IsUsed()</code> , <code>GetLength()</code> , <code>GetResX()</code> , <code>GetResY()</code>	TOF hits: matched to a global track, track length from the target [cm], residuals track – hit [cm]. SiBT hits: residuals to the beam track. Otherwise: false, -1, -1000	Both
<code>GetTimeStamp()</code>	TOF hits: time of flight [ns] (Section 6.5). Strip hits: -1 (not used)	Both
<code>GetDetectorID()</code> , <code>GetDetId()</code>	TOF hits: volume UID and <code>DetectorId</code> (Section 6.5). Strip hits: not filled (0)	Both
<code>GetType()</code> (fType)	Strip hits. Sim: 1 – hit matched to an MC point, 0 – fake (combinatorial) hit. Exp: -1 (undefined)	Sim
<code>GetNLinks()</code> , <code>GetLink(i)</code>	FairLinks to MC truth. Strip hits: links to the contributing MC points (index = MC point index, weight = contribution). TOF hits: type 1 – MC point, type 2 – MC track index, type 4 – detector UID. Exp: no links	Sim
<code>GetFlag()</code>	Not used (true for all hits)	–

6.2 Hits of the strip detectors

Branch	Class	Detector	Avail.
<code>BmnVSPHit</code>	<code>BmnVSPHit</code>	Vertex Silicon Plane	Both
<code>BmnSiliconHit</code>	<code>BmnSiliconHit</code>	Forward Silicon Detector (FSD)	Both
<code>BmnGemStripHit</code>	<code>BmnGemStripHit</code>	GEM tracker	Both
<code>BmnCSCHit</code>	<code>BmnCSCHit</code>	Cathode Strip Chambers	Both
<code>BmnSiBTHit</code>	<code>BmnSiBTHit</code>	Silicon Beam Tracker, upstream of the target	Exp

Station and module numbers start from 0. The number of stations and modules is defined by the detector configuration of the run period.

The strip hit classes provide getters for the characteristics of the lower and upper strip layers:

Getter (member)	Description	Source
<code>GetStripPositionInLowerLayer()</code> , <code>GetStripPositionInUpperLayer()</code>	Cluster positions in the two layers [strip units]	SiBT: stored in the hit; VSP, Silicon, GEM, CSC: clusters
<code>GetStripTotalSignalInLowerLayer()</code> , <code>GetStripTotalSignalInUpperLayer()</code>	Total cluster signals in the two layers	SiBT: stored in the hit; VSP, Silicon, GEM, CSC: clusters
<code>GetClusterSizeInLowerLayer()</code> , <code>GetClusterSizeInUpperLayer()</code>	Number of strips in the clusters	SiBT: stored in the hit; VSP, Silicon, GEM, CSC: clusters
<code>GetSignalDiff()</code> (fNormSignalDiff)	Normalised signal difference of the two layers (GEM, CSC, SiBT)	Not filled
<code>GetEnergyLoss()</code> (fELoss), <code>GetPdg()</code>	Energy loss; PDG code (VSP, Silicon)	Not filled

Note: VSP, Silicon, GEM and CSC hits do not store the layer characteristics: the getters take them from the clusters of the hit (`GetLowerClusterIndex()`, `GetUpperClusterIndex()`) in the branches `Bmn<Det>LowerCluster` and `Bmn<Det>UpperCluster` of the current event (Section 6.3); the strip position is the mean position of the cluster. The clusters are obtained via `FairRootManager`, so these getters work in FairRoot tasks, e.g. in `run_ana_bmn.C` (Section 9.2). When the DST tree is read directly (Section 9.3), they return the default values 0, 0 and -1: read the cluster branches in this case. For a pseudo hit (one-layer hit) the upper-layer getters return the default values.

6.3 Strip clusters (StripCluster)

The branches `Bmn<Det>UpperCluster` and `Bmn<Det>LowerCluster` (for VSP, Silicon, GEM and CSC) contain all reconstructed strip clusters, including clusters that are not part of any hit. Upper clusters have type 1, lower clusters type 0. The fields of `StripCluster` are public members:

Member (getter)	Description
<code>Station (GetStation())</code> , <code>Module (GetModule())</code>	Station and module of the cluster, starting from 0
<code>fType (GetType())</code>	0 – lower layer, 1 – upper layer
<code>MeanPosition</code>	Cluster position after fitting [strip units]
<code>OriginPosition</code> , <code>PositionResidual</code>	Position of the central strip and residual from it
<code>TotalSignal</code>	Sum of the strip signals [ADC counts]; the basis of the dE/dx measurement
<code>Width (GetWidth())</code>	Number of strips in the cluster
<code>Strips</code> , <code>Signals</code>	<code>vector<Int_t></code> strip numbers and <code>vector<Double_t></code> strip signals
<code>Error (GetError())</code>	Position error
<code>IsCorrect</code>	Not used (false)

6.4 Tracking hits: StsHit

Before the track finding, `BmnToCbmHitConverter` converts the VSP, Silicon and GEM hits into `CbmStsHit` objects with a common station numbering (pseudo hits are excluded). The hit coordinates are modified during the conversion:

- **Exp:** alignment corrections and, for the GEM in the magnetic field, a correction of the Lorentz shift are applied.
- **Sim:** MC corrections are applied (y shifted by +0.30 cm, smearing).
- In both cases the position errors are set to fixed values: GEM $\sigma_x = 0.028$ cm; Silicon and VSP $\sigma_x = 0.0029$ cm (VSP in experimental data: 0.0058 cm); $\sigma_y = 0.123$ cm for all.

Therefore the `StsHit` coordinates differ from those of the original `BmnVSPHit`, `BmnSiliconHit` and `BmnGemStripHit`. The `StsHit` coordinates are the ones used in the track fit.

Getter (member)	Description	Sim / Exp
<code>GetX()</code> , <code>GetY()</code> , <code>GetZ()</code> , <code>GetDx()</code> , <code>GetDy()</code>	Corrected position and fixed errors [cm]	Both
<code>GetSystemId()</code>	Detector: <code>kVSP = 8</code> , <code>kGEM = 2</code> , <code>kSILICON = 12</code> (bits 24–27 of <code>fDetectorID</code>)	Both
<code>GetStationNr()</code>	Tracking station, starting from 1 (bits 16–23): Run 9 – VSP 1, Silicon 2–5, GEM 6–13; without VSP (Run 8) – Silicon 1–4, GEM 5–11	Both
<code>GetSectorNr()</code>	GEM: 2·module + 1 + layer/2; Silicon and VSP: module + 1 (bits 4–15). <code>GetSensorNr()</code> is always 1	Both
<code>fDigiF</code> , <code>fDigiB</code> (public)	Index of the lower (F) and upper (B) cluster plus a detector offset: GEM 0 / 1 000 000, Silicon 2 000 000 / 3 000 000, VSP 4 000 000 / 5 000 000. Cluster index = value % 1 000 000	Both
<code>GetSignalDiv()</code>	Cluster signal asymmetry $(Q_{upper} - Q_{lower}) / (Q_{upper} + Q_{lower})$	Both
<code>GetTimeStamp()</code>	Not a time: cluster width asymmetry $(N_{upper} - N_{lower}) / (N_{upper} + N_{lower})$	Both
<code>GetStrip(0)</code> , <code>GetStrip(1)</code>	Mean positions of the lower and upper clusters [strip units]	Both
<code>GetRefIndex()</code>	Sim: MC point index. Exp: $-(\text{index of the original hit}) - 1$, i.e. original index = $-\text{GetRefIndex}() - 1$ in the hit branch given by <code>GetSystemId()</code>	Both
<code>GetNLinks()</code> , <code>GetLink(i)</code>	MC links copied from the original hit	Sim
<code>GetStatLayer()</code> , <code>GetPosSensorX/Y()</code> , <code>GetCovXY()</code>	Not used (0)	–

6.5 TOF-400 and TOF-700 hits

`BmnTof400Hit` and `BmnTof700Hit` contain `BmnTofHit` objects (derived from `BmnHit` without additional members). A hit is reconstructed in a strip of a multigap resistive plate chamber from the signals at both strip ends; the position is the crossing point along the strip.

Getter (member)	Description	Sim / Exp
<code>GetX()</code> , <code>GetY()</code> , <code>GetZ()</code>	Hit position [cm]; Exp: errors $\sigma_x = 0.5$ cm, $\sigma_y = 0.36$ cm	Both
<code>GetTimeStamp()</code>	Time of flight [ns]. Exp: hit time minus the T_0 time (Run 9: <code>BmnTrigInfo.T0</code>) with calibration and slewing corrections. Sim: MC time since the event start, smeared	Both
<code>GetModule()</code> , <code>GetStation()</code>	Chamber (plane) number and strip number, both starting from 0	Both
<code>GetDetectorID()</code>	Volume UID: $(\text{plane} + 1) \ll 8 \mid (\text{strip} + 1)$; decode with <code>BmnTOF1Point::GetModule(uid)</code> and <code>BmnTOF1Point::GetStrip(uid)</code>	Both
<code>GetDetId()</code>	<code>kTOF1 (3)</code> for TOF-400, <code>kTOF701 (6)</code> for TOF-700	Both
<code>IsUsed()</code> , <code>GetLength()</code> , <code>GetResX()</code> , <code>GetResY()</code>	Matched to a global track; track length from the target [cm]; residuals track – hit after a momentum-dependent correction [cm]	Both
<code>GetRefIndex()</code>	Sim: MC point index. Exp: packed indices of the digits at the two strip ends	Both
<code>GetUpperClusterIndex()</code> , <code>GetLowerClusterIndex()</code>	Exp: indices of the left-end and right-end digits (the digits are not stored). Sim: undefined	Exp
<code>GetLink(i)</code>	Type 1 – MC point, type 2 – MC track index, type 4 – volume UID	Sim

Caution: In experimental DSTs the time of flight, and therefore β and m^2 of the global tracks, depend on the TOF time calibration (T_0 , slewing and strip corrections) applied in the reconstruction. Check the time of flight distribution of the production before using β or m^2 .

7 Forward detectors

The forward detectors are stored as single event objects: one object per event with all modules, cells or strips of the detector.

7.1 FHCaI: FHCaIEvent

Getter	Description
<code>BmnFHCaIEvent::GetTotalEnergy()</code>	Total energy deposited in the Forward Hadron Calorimeter [MIP]
<code>BmnFHCaIEvent::GetTotalModules()</code>	Number of modules
<code>BmnFHCaIEvent::GetModule(id)</code>	Module <code>BmnFHCaIModule</code> with <code>id = 1 ... GetTotalModules()</code>
<code>BmnFHCaIModule::GetModuleId(), GetEnergy()</code>	Module number and module energy [MIP]
<code>BmnFHCaIModule::GetNsections(), GetSectionEnergy(sec)</code>	Number of longitudinal sections and section energy, <code>sec = 1 ... N</code>
<code>BmnFHCaIModule::GetX(), GetY(), GetZ()</code>	Module centre in the global coordinate system [cm]

7.2 Scintillation Wall: ScWallEvent

Getter	Description
<code>BmnScWallEvent::GetCell(id)</code>	Cell <code>BmnScWallCell</code> with <code>id = 1 ... 174</code> ; all 174 cells are always stored
<code>BmnScWallCell::GetCellId()</code>	Cell number
<code>BmnScWallCell::GetSignal()</code>	Signal in the cell, calibrated in units of Z^2 of the crossing particle; 0 – no signal
<code>BmnScWallCell::GetX(), GetY(), GetZ()</code>	Cell centre [cm]

7.3 Forward hodoscope: HodoEvent

Getter	Description
<code>BmnHodoEvent::GetTotalSignal()</code>	Sum of the strip signals in units of Z^2 (the Xe beam peak corresponds to $54^2 \approx 2916$); input of the centrality classification
<code>BmnHodoEvent::GetTotalSignalIntegral()</code>	Sum of the strip signal integrals
<code>BmnHodoEvent::GetPileUpStatus()</code>	<code>BmnHodoStrip::PileUp</code> : No (0), Resolved (1), Unresolved (2); the maximum over the strips
<code>BmnHodoEvent::GetStrips(), GetStrip(id)</code>	<code>TClonesArray</code> of <code>BmnHodoStrip</code> , one entry per strip; in experimental data additional read-out channels can appear as entries with other identifiers
<code>BmnHodoStrip::GetStripId(), GetSignal(), GetIntegral(), GetPileUpStatus()</code>	Strip number, signal, integral, pile-up status
<code>BmnHodoStrip::GetX(), GetY(), GetZ()</code>	Strip position [cm]
<code>BmnHodoStrip::GetDigits()</code>	Not stored in the DST (empty)

7.4 Neutron detector: NdetEvent

Getter	Description
<code>BmnNdetEvent::GetCells(), GetTotalCells()</code>	<code>TClonesArray</code> of fired cells <code>BmnNdetCell</code> and their number
<code>BmnNdetEvent::GetCell(address), GetCell(arm, row, column, layer)</code>	Cell by address or by indices
<code>BmnNdetCell::GetAddress()</code>	Packed cell address; decode with <code>GetArmId(), GetRowId(), GetColumnId(), GetLayerId()</code>
<code>BmnNdetCell::GetSignal(), GetTime()</code>	Cell signal and time [ns]
<code>BmnNdetCell::GetX(), GetY(), GetZ()</code>	Cell position [cm]

8 Monte Carlo information (simulated DSTs)

8.1 BmnMCInfo. and MC tracks

BmnMCInfoDST keeps the generator-level information of the event and all MC tracks of the simulation stack. The MC points of the detectors are **not** stored in the DST.

Getter (member)	Description
GetB()	Impact parameter [fm]
GetX(), GetY(), GetZ()	Generated primary vertex [cm]
GetRotX(), GetRotY()	Rotation around the x and y axes (beam tilt) [rad]
GetRotZ()	Rotation around the z axis (reaction plane angle) [rad]
GetMCTracks()	TClonesArray of CbmMCTrack: primary particles and secondaries stored by the simulation. The order is the same as in the simulation file, so MC track indices of the matching objects point into this array

Fields of CbmMCTrack:

Getter (member)	Description
GetPdgCode()	PDG code; nuclei have codes 100ZZZAAAI (e.g. 1000010020 for the deuteron)
GetMotherId()	Index of the mother track in the same array; -1 for primary particles
GetPx(), GetPy(), GetPz(), GetMomentum(TVector3&)	Momentum at the start vertex [GeV/c]
GetStartX(), GetStartY(), GetStartZ(), GetStartT()	Start vertex [cm] and time [ns]
GetP(), GetPt(), GetRapidity(), GetEnergy(), GetMass()	Derived kinematics; the mass is taken from TDatabasePDG
Get4Momentum(TLorentzVector&)	Four-momentum
GetNPoints(DetectorId)	Number of MC points of the track in a detector (bit-packed member fNPoints)
GetPolar(i)	Polarisation components, i = 0, 1, 2

8.2 Matching of reconstructed tracks to MC tracks

BmnMatchRecoToMC links every global track to the MC track that produced most of its hits:

Object	Getter	Description
BmnGlobalTrackMatch[i] (BmnTrackMatch)	GetNofLinks(), GetLink(j)	All MC tracks contributing to the hits of global track i: BmnLink with GetIndex() = MC track index and GetWeight() = number of hits
	GetMatchedLink()	Link with the largest weight
	GetNofTrueHits(), GetNofWrongHits(), GetTrueOverAllHitsRatio()	Number of hits from the matched MC track, other hits, and their ratio
BmnGlobalTrack	GetRefIndex()	Index of the matched MC track if the true-hit ratio is ≥ 0.6 , otherwise -1
StsTrackMatch[i] (CbmTrackMatch)	GetMCTrackId(), GetNofTrueHits(), GetNofWrongHits(), GetNofFakeHits(), GetNofMCTracks()	Matching of inner track i of StsVector done by BmnStsMatchTracks
CbmStsTrack	GetTrkID()	Same MC track index as BmnGlobalTrack::GetRefIndex()

On the hit level, `GetRefIndex()` gives the MC point index, and the FairLinks point to the contributing MC points. For TOF hits the link of type 2 gives the MC track index directly (Section 6.1).

8.3 NdetSurfacePoint

`NdetSurfacePoint` contains `BmnNdetSurfacePoint` objects: MC points of particles crossing the surfaces of the neutron detector, for efficiency studies. The class extends `FairMCPPoint` (track index, position, momentum, time, track length, energy loss) with `fPdg`, `fArmID`, `fPlaneID` and `fDirectionID`.

9 Physics analysis of reconstructed data

There are two ways to analyse DST files. The **recommended** way is to implement every analysis as a FairRoot task and to run it with the analysis macro `macro/run/run_ana_bmn.C` (Section 9.2). Reading the DST tree directly in a ROOT macro (Section 9.3) is an alternative for quick checks and simple studies.

9.1 Environment

The DST classes are BmnRoot and FairRoot classes, so the BmnRoot environment must be loaded before ROOT is started; the class dictionaries are then found automatically:

```
source <bmnroot build directory>/config.sh
```

9.2 Recommended: analysis tasks in run_ana_bmn.C

`run_ana_bmn.C` is the main macro for the physics analysis of simulated and experimental DSTs. It sets up `FairRunAna` with the DST as input and executes a list of analysis tasks for every event. The macro takes care of the common steps, so that a task contains only the analysis itself:

- the input can be one DST file, a text file with a list of DST files, or a ROOT file with a `TChain` of DST files; all files are processed as one data set;
- the magnetic field and the run information (period, run, beam, target, energy) are taken from the first DST file, and the geometry is restored from its `FairGeoParSet` container;
- the Kalman filter (`CbmKF`) is initialised, so that the tasks can extrapolate tracks through the detector material and the magnetic field;
- the results of the tasks (histograms, trees, new branches) are written to the output file.

Arguments of the macro:

Argument	Description
<code>dstFileName</code>	Input: DST file, text file with a list of DST files, or ROOT file with a <code>TChain</code> of DST files
<code>anaFileName</code>	Output file with the analysis results
<code>nStartEvent</code>	First event to process, starting from 0
<code>nEvents</code>	Number of events to process; 0 – all events of the input after <code>nStartEvent</code>

```
root -b -q '$VMCWORKDIR/macro/run/run_ana_bmn.C("bmndst.root", "bmnan.root", 0, 0)'
```

Each analysis is a class derived from `FairTask` with three main methods:

- `Init()` – called once before the event loop: get the DST branches with `FairRootManager::GetObject()` (branch names as in Section 2.2), book histograms, read parameters;
- `Exec()` – called for every event: the objects obtained in `Init()` contain the data of the current event;
- `Finish()` – called once after the event loop: write the results to the output file.

The analysis tasks are kept in the `physics` library of BmnRoot. The class `BmnAnalysisTask` (`physics/BmnAnalysisTask.h` and `.cxx`) is a template to start from, and `BmnTwoParticleDecay`

([physics/decay](#)) is an example of a complete analysis task. A new task is added to [physics/CMakeLists.txt](#) and [physics/PhysicsLinkDef.h](#). The example below fills the transverse momentum spectrum of the primary global tracks (for brevity the methods are defined in the class body):

```
// BmnPtAnalysis: example analysis task - transverse momentum spectrum of primary global tracks
class BmnPtAnalysis : public FairTask
{
public:
    BmnPtAnalysis()
        : FairTask("BmnPtAnalysis")
    {}

    InitStatus Init() override
    {
        // get the DST branches from the input of FairRunAna
        FairRootManager* ioman = FairRootManager::Instance();
        fGlobalTracks = (TClonesArray*)ioman->GetObject("BmnGlobalTrack");
        fInnerTracks = (TClonesArray*)ioman->GetObject("StsVector");
        fVertex = (CbmVertex*)ioman->GetObject("MpdVertex.");
        if (!fGlobalTracks || !fInnerTracks || !fVertex) {
            LOG(error) << "BmnPtAnalysis::Init(): DST branches not found, the task is deactivated";
            return kERROR;
        }
        fHistPt = new TH1D("hPt", "p_{T} of primary tracks;p_{T}, GeV/c;tracks", 100, 0., 3.);
        return kSUCCESS;
    }

    void Exec(Option_t* /*option*/) override
    {
        if (fVertex->GetNTracks() < 2)
            return;
        for (Int_t i = 0; i < fGlobalTracks->GetEntriesFast(); i++) {
            BmnGlobalTrack* track = (BmnGlobalTrack*)fGlobalTracks->UncheckedAt(i);
            CbmStsTrack* inner = (CbmStsTrack*)fInnerTracks->UncheckedAt(track->GetGemTrackIndex());
            if (!inner->FromVertex()) // keep the tracks used in the primary vertex fit
                continue;
            fHistPt->Fill(track->GetPt());
        }
    }

    void Finish() override
    {
        // write the results to the output file of run_ana_bmn.C
        FairRootManager::Instance()->GetSink()->WriteObject(fHistPt, "hPt");
    }

private:
    TClonesArray* fGlobalTracks = nullptr;
    TClonesArray* fInnerTracks = nullptr;
    CbmVertex* fVertex = nullptr;
    TH1D* fHistPt = nullptr;

    ClassDefOverride(BmnPtAnalysis, 1);
};
```

The task is created in the “Analysis Tasks” block of [run_ana_bmn.C](#):

```
// ===== //
// == Analysis Tasks == //
// ===== //
CbmKF* kalman_filter = new CbmKF();
fRunAna->AddTask(kalman_filter);

BmnPtAnalysis* pt_analysis = new BmnPtAnalysis();
fRunAna->AddTask(pt_analysis);
```

Tip: Several analyses can run in one pass over the data: add one task for each of them. The tasks are executed in the order in which they are added, so a task can use objects that a previous task registered with `FairRootManager::Register()`.

9.3 Alternative: reading a DST file directly

For quick checks the DST tree can also be read directly in a ROOT macro. In this case the macro itself has to handle the input files and, if needed, the magnetic field and the geometry. The macro below works with simulated and experimental DSTs. It prints the run information and the vertex, and for the first tracks the charge, momentum, vertex association, TOF-400 information and, if available, the matched MC track.

```
// read_dst_example.C: loop over DST events and print basic information on vertices and global tracks
// works with simulated and experimental DSTs; MC information is read only if it is present in the file
void read_dst_example(TString dstFileName = "bmndst.root", Int_t nPrint = 3)
{
    // run-level information is stored once per file
    TFile* file = TFile::Open(dstFileName);
    DstRunHeader* runHeader = (DstRunHeader*)file->Get("DstRunHeader");
    cout << "Period " << runHeader->GetPeriodId() << ", run " << runHeader->GetRunId() << ": "
        << runHeader->GetBeam() << " + " << runHeader->GetTarget() << " at " << runHeader->GetBeamEnergy()
        << " AGeV, field scale " << runHeader->GetFieldScale() << endl;

    TTree* tree = (TTree*)file->Get("bmndata");
    tree->SetBranchStatus("*Cluster*", 0); // skip the heavy cluster branches, they are not used here

    DstEventHeader* eventHeader = nullptr;
    CbmVertex* vertex = nullptr;
    TClonesArray *globalTracks = nullptr, *innerTracks = nullptr, *tof400Hits = nullptr;
    tree->SetBranchStatus("DstEventHeader.", &eventHeader);
    tree->SetBranchStatus("MpdVertex.", &vertex);
    tree->SetBranchStatus("BmnGlobalTrack", &globalTracks);
    tree->SetBranchStatus("StsVector", &innerTracks);
    tree->SetBranchStatus("BmnTof400Hit", &tof400Hits);

    // MC information exists only in simulated DSTs
    BmnMCInfoDst* mcInfo = nullptr;
    Bool_t isSimulation = (tree->GetBranch("BmnMCInfo.") != nullptr);
    if (isSimulation)
        tree->SetBranchStatus("BmnMCInfo.", &mcInfo);

    for (Long64_t iEvent = 0; iEvent < TMath::Min(tree->GetEntries(), (Long64_t)nPrint); iEvent++) {
        tree->GetEntry(iEvent);
        printf("Event %u: vertex (%.2f, %.2f, %.2f) cm from %d tracks, %d global tracks\n", eventHeader-
            >GetEventId(),
            vertex->GetX(), vertex->GetY(), vertex->GetZ(), vertex->GetNTracks(), globalTracks-
            >GetEntriesFast());
        if (vertex->GetNTracks() < 2)
            continue;

        for (Int_t iTrack = 0; iTrack < TMath::Min(globalTracks->GetEntriesFast(), 3); iTrack++) {
            BmnGlobalTrack* track = (BmnGlobalTrack*)globalTracks->UncheckedAt(iTrack);
            CbmStsTrack* inner = (CbmStsTrack*)innerTracks->UncheckedAt(track->GetGemTrackIndex());

            FairTrackParam* par = track->GetParamFirst(); // at the z of MpdVertex
            Double_t p = 1. / TMath::Abs(par->GetQp()); // momentum for |Z| = 1, rigidity p/|Z| otherwise
            Int_t charge = (par->GetQp() > 0) ? +1 : -1;
            Double_t pz = p / TMath::Sqrt(1. + par->GetTx() * par->GetTx() + par->GetTy() * par->GetTy());
            Double_t pt = pz * TMath::Sqrt(par->GetTx() * par->GetTx() + par->GetTy() * par->GetTy());

            printf(" track %d: q = %d, p = %.3f, pt = %.3f GeV/c, hits = %d, in vertex = %d, DCA = %.2f cm",
                iTrack,
                charge, p, pt, inner->GetNStsHits(), inner->FromVertex() ? 1 : 0, track->GetDCAInVertex());
            if (track->GetTof1HitIndex() >= 0) {
                BmnTofHit* hit = (BmnTofHit*)tof400Hits->UncheckedAt(track->GetTof1HitIndex());
                printf(" TOF400: t = %.2f ns, beta = %.3f, m2 = %.3f", hit->GetTimeStamp(), track->GetBeta(1),
                    track->GetMass2(1));
            }
            if (isSimulation && track->GetRefIndex() >= 0) {
                CbmMCTrack* mcTrack = (CbmMCTrack*)mcInfo->GetMCTracks()->UncheckedAt(track->GetRefIndex());
                printf(" MC: pdg = %d, p = %.3f", mcTrack->GetPdgCode(), mcTrack->GetP());
            }
            printf("\n");
        }
    }
}
```

```
root -l 'read_dst_example.C("bmndst.root")'
```

9.4 Quick look with TTree::Draw

Member names can be used directly in TTree::Draw expressions:

```
TFile* f = TFile::Open("bmndst.root");
TTree* t = (TTree*)f->Get("bmndata");
t->Draw("MpdVertex.fZ", "MpdVertex.fNTracks > 1");           // vertex z
t->Draw("1./BmnGlobalTrack.fParamFirst.fQp");                 // signed momentum
t->Draw("BmnGlobalTrack.fBeta400", "BmnGlobalTrack.fTof1Hit >= 0"); // beta from TOF-400
t->Draw("HodoEvent.fTotalSignal:FHCalEvent.fTotalEnergy", "", "colz"); // centrality plane
```

9.5 Reading only the needed branches

Most of the data volume is in the hit and cluster branches (Appendix E). When the tree is read directly, disable the branches you do not need before the event loop:

```
tree->SetBranchStatus("*Cluster*", 0); // all strip cluster branches
tree->SetBranchStatus("StsHit*", 0);   // tracking hits
tree->SetBranchStatus("Bmn*Hit*", 0);   // detector hits (switch on the TOF hits again if needed)
tree->SetBranchStatus("BmnTof*", 1);
```

9.6 Further examples

- [macro/howTo/readDST.C](#) – a basic example of reading a DST directly; it also shows how to obtain the magnetic field of the run with `BmnFunctionAna::GetDSTField()`.

Appendix A. Summary of the DST content

The table lists all objects of a DST file with a short description and the main methods of their classes. The objects stored once per file come first, followed by the branches of the event tree `bmndata`, written as `bmndata.<branch>`. For `TClonesArray` branches the methods are those of the array elements, which are accessed with `GetEntriesFast()` and `UncheckedAt(i)`. The notation `GetX/Y/Z()` stands for `GetX()`, `GetY()` and `GetZ()`. Objects marked *Sim only* or *Exp only* exist in one type of DST only; the details are given in Sections 2–8.

Object in the DST file	Content	Main methods
File objects: FairRoot bookkeeping		
<code>bmnroot</code>	FairRoot folder structure: which branch belongs to which detector folder (<code>TFolder</code>)	<code>FindObjectAny(name)</code> , <code>GetListOfFolders()</code>
<code>BranchList</code>	Names of all branches known during the reconstruction, including input branches that are not stored in the DST (<code>TList</code> of <code>TObjString</code>)	<code>GetEntries()</code> , <code>At(i)->GetName()</code>
<code>TimeBasedBranchList</code>	FairRoot list of the time-based branches (<code>TList</code>), not used in BmnRoot	<code>GetEntries()</code>
File objects: parameter containers		
<code>BmnFieldPar</code>	<i>Sim only</i> : magnetic field parameters of the simulation	<code>GetType()</code> , <code>GetScale()</code> , <code>MapName(name)</code> , <code>GetPositionX/Y/Z()</code> , <code>IsFieldOff()</code>
<code>FairBaseParSet</code>	<i>Sim only</i> : simulation run parameters; the beam momentum field holds the beam energy per nucleon, the description holds “beam;target”	<code>GetBeamMom()</code> , <code>getDescription()</code> , <code>GetRndSeed()</code>
<code>FairGeoParSet</code>	Geometry of the reconstruction (<code>TGeoManager</code>); the description holds the geometry tag	<code>GetGeometry()</code> , <code>getDescription()</code> , <code>GetGeoNodes()</code>
<code>FairGeoPassivePar</code>	<i>Sim only</i> : passive geometry parameters of the simulation	<code>GetGeoPassiveNodes()</code> , <code>GetGeoSensitiveNodes()</code>
<code><run ID></code>	Runtime database bookkeeping (<code>FairRtdbRun</code>); the key name is the run ID (the period number for MC)	<code>getRunId()</code> , <code>getParVersions()</code>
File objects: headers		
<code>DstRunHeader</code>	Run information: period, run, start and finish times, beam, target, beam energy, magnetic field (<code>DstRunHeader</code>)	<code>GetPeriodId()</code> , <code>GetRunId()</code> , <code>GetStartTime()</code> , <code>GetFinishTime()</code> , <code>GetBeam()</code> , <code>GetBeamA()</code> , <code>GetBeamZ()</code> , <code>GetBeamEnergy()</code> , <code>GetTarget()</code> , <code>GetTargetA()</code> , <code>GetTargetZ()</code> , <code>GetFieldType()</code> , <code>GetFieldScale()</code> , <code>GetFieldName()</code>
<code>FileHeader</code>	FairRoot file header (<code>FairFileHeader</code>)	<code>GetRunId()</code> , <code>GetListOfTasks()</code>
Event tree bmndata: event information		
<code>bmndata.BmnTrigInfo.</code>	Trigger input signals, amplitudes of the beam counters and of the fragment detector, T0 start time, trigger digits (<code>BmnTrigInfoDst</code>)	<code>GetInputSignalsAR()</code> , <code>GetInputSignalsBR()</code> , <code>IsTriggerBitTrueAR(bit)</code> , <code>IsTriggerBitTrueBR(bit)</code> , <code>GetInputSignalsVector()</code> , <code>GetT0_time()</code> , <code>GetT0_amp()</code> , <code>GetVCamp()</code> , <code>GetBC0Amp()</code> , <code>GetBC1Amp()</code> , <code>GetBC1Integral()</code> , <code>GetBC2Amp()</code> , <code>GetFDAmp()</code> , <code>GetBDMult()</code> , <code>GetSiMDMult()</code> , <code>GetBC0Digits()</code> , <code>GetBC1Digits()</code> , <code>GetBC2Digits()</code> , <code>GetBDDigits()</code> , <code>GetFDDigits()</code>
<code>bmndata.DstEventHeader.</code>	Event header: run and event numbers, period, event time, impact parameter (<code>DstEventHeader</code>)	<code>GetRunId()</code> , <code>GetEventId()</code> , <code>GetPeriodId()</code> , <code>GetEventTime()</code> , <code>GetEventTimeTS()</code> , <code>GetB()</code> , <code>GetTriggerType()</code> , <code>GetZ2in()</code> , <code>GetZ2out()</code> , <code>GetADCin()</code> , <code>GetADCout()</code>
<code>bmndata.EventCentrality</code>	Centrality class of the event from the FHCAL energy and the hodoscope signal (<code>BmnEventCentrality</code>)	<code>GetClass()</code> , <code>GetProbability()</code> , <code>GetSoftCentrality()</code>
Event tree bmndata: strip clusters		
<code>bmndata.BmnCSCLowerCluster</code>	Lower layer () strip clusters of the CSC (<code>TClonesArray</code> of <code>StripCluster</code>)	<code>GetStation()</code> , <code>GetModule()</code> , <code>GetType()</code> , <code>GetWidth()</code> , <code>GetError()</code> ; members <code>MeanPosition</code> , <code>TotalSignal</code> , <code>Strips</code> , <code>Signals</code>

Object in the DST file	Content	Main methods
bmndata. BmnCSCUpperCluster	Upper layer (\\ or //) strip clusters of the CSC (TClonesArray of StripCluster)	GetStation(), GetModule(), GetType(), GetWidth(), GetError(); members MeanPosition, TotalSignal, Strips, Signals
bmndata. BmnGemLowerCluster	Lower layer () strip clusters of the GEM (TClonesArray of StripCluster)	GetStation(), GetModule(), GetType(), GetWidth(), GetError(); members MeanPosition, TotalSignal, Strips, Signals
bmndata. BmnGemUpperCluster	Upper layer (\\ or //) strip clusters of the GEM (TClonesArray of StripCluster)	GetStation(), GetModule(), GetType(), GetWidth(), GetError(); members MeanPosition, TotalSignal, Strips, Signals
bmndata. BmnSiliconLowerCluster	Lower layer () strip clusters of the Silicon detector (TClonesArray of StripCluster)	GetStation(), GetModule(), GetType(), GetWidth(), GetError(); members MeanPosition, TotalSignal, Strips, Signals
bmndata. BmnSiliconUpperCluster	Upper layer (\\ or //) strip clusters of the Silicon detector (TClonesArray of StripCluster)	GetStation(), GetModule(), GetType(), GetWidth(), GetError(); members MeanPosition, TotalSignal, Strips, Signals
bmndata. BmnVSPLowerCluster	Lower layer () strip clusters of the VSP (TClonesArray of StripCluster)	GetStation(), GetModule(), GetType(), GetWidth(), GetError(); members MeanPosition, TotalSignal, Strips, Signals
bmndata. BmnVSPUpperCluster	Upper layer (\\ or //) strip clusters of the VSP (TClonesArray of StripCluster)	GetStation(), GetModule(), GetType(), GetWidth(), GetError(); members MeanPosition, TotalSignal, Strips, Signals
Event tree bmndata: hits		
bmndata.BmnCSCHit	Hits of the Cathode Strip Chambers (TClonesArray of BmnCSCHit)	GetX/Y/Z(), GetDx/Dy/Dz(), GetStation(), GetModule(), GetLowerClusterIndex(), GetUpperClusterIndex(), GetStripTotalSignalInLowerLayer(), GetStripTotalSignalInUpperLayer(), GetClusterSizeInLowerLayer(), GetClusterSizeInUpperLayer(), GetStripPositionInLowerLayer(), GetStripPositionInUpperLayer(), IsPseudo(), GetRefIndex()
bmndata. BmnGemStripHit	GEM hits (TClonesArray of BmnGemStripHit)	GetX/Y/Z(), GetDx/Dy/Dz(), GetStation(), GetModule(), GetLowerClusterIndex(), GetUpperClusterIndex(), GetStripTotalSignalInLowerLayer(), GetStripTotalSignalInUpperLayer(), GetClusterSizeInLowerLayer(), GetClusterSizeInUpperLayer(), GetStripPositionInLowerLayer(), GetStripPositionInUpperLayer(), IsPseudo(), GetRefIndex()
bmndata.BmnSiBTHit	<i>Exp only:</i> hits of the Silicon Beam Tracker upstream of the target (TClonesArray of BmnSiBTHit)	GetX/Y/Z(), GetDx/Dy/Dz(), GetStation(), GetModule(), GetLowerClusterIndex(), GetUpperClusterIndex(), GetStripTotalSignalInLowerLayer(), GetStripTotalSignalInUpperLayer(), GetClusterSizeInLowerLayer(), GetClusterSizeInUpperLayer(), GetStripPositionInLowerLayer(), GetStripPositionInUpperLayer(), IsPseudo(), GetRefIndex(), GetResX/Y()
bmndata. BmnSiliconHit	Hits of the Forward Silicon Detector (TClonesArray of BmnSiliconHit)	GetX/Y/Z(), GetDx/Dy/Dz(), GetStation(), GetModule(), GetLowerClusterIndex(), GetUpperClusterIndex(), GetStripTotalSignalInLowerLayer(), GetStripTotalSignalInUpperLayer(), GetClusterSizeInLowerLayer(), GetClusterSizeInUpperLayer(), GetStripPositionInLowerLayer(), GetStripPositionInUpperLayer(), IsPseudo(), GetRefIndex()
bmndata.BmnToF400Hit	TOF-400 hits (TClonesArray of BmnToFHit)	GetX/Y/Z(), GetTimeStamp() (time of flight), GetModule(), GetStation() (strip), GetDetectorID(), IsUsed(), GetLength(), GetResX/Y(), GetRefIndex()

Object in the DST file	Content	Main methods
<code>bmndata.BmnTof700Hit</code>	TOF-700 hits (TClonesArray of BmnTofHit)	<code>GetX/Y/Z()</code> , <code>GetTimeStamp()</code> (time of flight), <code>GetModule()</code> , <code>GetStation()</code> (strip), <code>GetDetectorID()</code> , <code>IsUsed()</code> , <code>GetLength()</code> , <code>GetResX/Y()</code> , <code>GetRefIndex()</code>
<code>bmndata.BmnVSPHit</code>	Hits of the Vertex Silicon Plane (TClonesArray of BmnVSPHit)	<code>GetX/Y/Z()</code> , <code>GetDx/Dy/Dz()</code> , <code>GetStation()</code> , <code>GetModule()</code> , <code>GetLowerClusterIndex()</code> , <code>GetUpperClusterIndex()</code> , <code>GetStripTotalSignalInLowerLayer()</code> , <code>GetStripTotalSignalInUpperLayer()</code> , <code>GetClusterSizeInLowerLayer()</code> , <code>GetClusterSizeInUpperLayer()</code> , <code>GetStripPositionInLowerLayer()</code> , <code>GetStripPositionInUpperLayer()</code> , <code>IsPseudo()</code> , <code>GetRefIndex()</code>
<code>bmndata.StsHit</code>	Tracking hits: VSP, Silicon and GEM hits with the corrected positions used in the track fit (TClonesArray of CbmStsHit)	<code>GetX/Y/Z()</code> , <code>GetDx/Dy()</code> , <code>GetSystemId()</code> , <code>GetStationNr()</code> , <code>GetSectorNr()</code> , <code>GetSignalDiv()</code> , <code>GetStrip(side)</code> , <code>GetRefIndex()</code> ; members <code>fDigiF</code> , <code>fDigiB</code> (cluster indices)
Event tree bmndata: tracks		
<code>bmndata.BmnBeamTrack</code>	<i>Exp only</i> : beam ion tracks in the Silicon Beam Tracker (TClonesArray of BmnTrack)	<code>GetNHits()</code> , <code>GetHitIndex(i)</code> , <code>GetParamFirst()</code> , <code>GetParamLast()</code> , <code>GetChi2()</code> , <code>GetB()</code>
<code>bmndata.StsTrackMatch</code>	<i>Sim only</i> : MC matching of the inner tracks (TClonesArray of CbmTrackMatch)	<code>GetMCTrackId()</code> , <code>GetNofTrueHits()</code> , <code>GetNofWrongHits()</code> , <code>GetNofFakeHits()</code> , <code>GetNofMCTracks()</code>
<code>bmndata.StsVector</code>	Inner tracks in the VSP, Silicon and GEM detectors (TClonesArray of CbmStsTrack)	<code>GetNStsHits()</code> , <code>GetStsHitIndex(i)</code> , <code>GetParamFirst()</code> , <code>GetParamLast()</code> , <code>GetChi2()</code> , <code>GetNDF()</code> , <code>GetFlag()</code> , <code>FromVertex()</code> , <code>GetB()</code> , <code>GetPt()</code> , <code>GetEta()</code> , <code>GetTrkID()</code>
Event tree bmndata: vertices		
<code>bmndata.CombineVertex</code>	<i>Exp only</i> : primary vertices combined with the beam track, [0] from <code>PrimaryVertex.</code> , [1] from <code>MpdVertex.</code> (TClonesArray of CbmVertex)	<code>GetX/Y/Z()</code> , <code>GetChi2()</code> , <code>GetNDF()</code> , <code>GetNTracks()</code> , <code>GetCovariance(i, j)</code> , <code>GetTrackInds()</code>
<code>bmndata.MpdVertex.</code>	Primary vertex, the reference vertex of the global tracks (CbmVertex)	<code>GetX/Y/Z()</code> , <code>GetChi2()</code> , <code>GetNDF()</code> , <code>GetNTracks()</code> , <code>GetCovariance(i, j)</code> , <code>GetTrackInds()</code>
<code>bmndata.PrimaryVertex.</code>	Primary vertex of the CBM KF vertex finder (CbmVertex)	<code>GetX/Y/Z()</code> , <code>GetChi2()</code> , <code>GetNDF()</code> , <code>GetNTracks()</code> , <code>GetCovariance(i, j)</code> , <code>GetTrackInds()</code>
Event tree bmndata: global tracks		
<code>bmndata.BmnGlobalTrack</code>	Global tracks: inner tracks with matched TOF-400 and TOF-700 hits, β and m^2 (TClonesArray of BmnGlobalTrack)	<code>GetParamFirst()</code> , <code>GetParamLast()</code> , <code>GetP()</code> , <code>GetPx/Py/Pz()</code> , <code>GetPt()</code> , <code>GetEta()</code> , <code>GetTheta()</code> , <code>GetPhi()</code> , <code>GetGemTrackIndex()</code> , <code>GetTof1HitIndex()</code> , <code>GetTof2HitIndex()</code> , <code>GetBeta(tof)</code> , <code>GetMass2(tof)</code> , <code>GetDCAInVertex()</code> , <code>GetChi2InVertex()</code> , <code>GetNHits()</code> , <code>GetChi2()</code> , <code>GetNDF()</code> , <code>GetLength()</code> , <code>GetScWallCellId()</code> , <code>GetDedx()</code> , <code>GetQ()</code> , <code>GetRefIndex()</code>
<code>bmndata.BmnGlobalTrackMatch</code>	<i>Sim only</i> : MC matching of the global tracks (TClonesArray of BmnTrackMatch)	<code>GetNofLinks()</code> , <code>GetLink(i)</code> , <code>GetMatchedLink()</code> , <code>GetNofTrueHits()</code> , <code>GetNofWrongHits()</code> , <code>GetTrueOverAllHitsRatio()</code>
Event tree bmndata: forward detectors		
<code>bmndata.FHCalEvent</code>	Forward Hadron Calorimeter: total energy, modules and their sections (BmnFHCalEvent)	<code>GetTotalEnergy()</code> , <code>GetTotalModules()</code> , <code>GetModule(id)</code> ; <code>module</code> : <code>GetModuleId()</code> , <code>GetEnergy()</code> , <code>GetNsections()</code> , <code>GetSectionEnergy(sec)</code> , <code>GetX/Y/Z()</code>
<code>bmndata.HodoEvent</code>	Forward hodoscope: total signal in Z^2 units and strips (BmnHodoEvent)	<code>GetTotalSignal()</code> , <code>GetTotalSignalIntegral()</code> , <code>GetPileUpStatus()</code> , <code>GetStrips()</code> , <code>GetStrip(id)</code> ; <code>strip</code> : <code>GetStripId()</code> , <code>GetSignal()</code> , <code>GetIntegral()</code> , <code>GetPileUpStatus()</code> , <code>GetX/Y/Z()</code>
<code>bmndata.NdetEvent</code>	Neutron detector: fired cells (BmnNdetEvent)	<code>GetCells()</code> , <code>GetTotalCells()</code> , <code>GetCell(address)</code> ; <code>cell</code> : <code>GetAddress()</code> , <code>GetArmId()</code> , <code>GetRowId()</code> , <code>GetColumnId()</code> , <code>GetLayerId()</code> , <code>GetSignal()</code> , <code>GetTime()</code> , <code>GetX/Y/Z()</code>
<code>bmndata.ScWallEvent</code>	Scintillation Wall: 174 cells with their signals (BmnScWallEvent)	<code>GetCell(id)</code> ; <code>cell</code> : <code>GetCellId()</code> , <code>GetSignal()</code> , <code>GetX/Y/Z()</code>

Object in the DST file	Content	Main methods
Event tree bmndata: Monte Carlo information		
<code>bmndata.BmnMCInfo.</code>	<i>Sim only:</i> generator information of the event and MC tracks (<code>BmnMCInfoDst</code>)	<code>GetB()</code> , <code>GetX/Y/Z()</code> , <code>GetRotX/Y/Z()</code> , <code>GetMCTracks()</code> ; MC track: <code>GetPdgCode()</code> , <code>GetMotherId()</code> , <code>GetPx/Py/Pz()</code> , <code>GetP()</code> , <code>GetPt()</code> , <code>GetRapidity()</code> , <code>GetStartX/Y/Z/T()</code> , <code>GetNPoints(detector)</code>
<code>bmndata.NdetSurfacePoint</code>	<i>Sim only:</i> MC points of particles crossing the neutron detector surfaces (<code>TClonesArray</code> of <code>BmnNdetSurfacePoint</code>)	<code>GetTrackID()</code> , <code>GetX/Y/Z()</code> , <code>GetPx/Py/Pz()</code> , <code>GetTime()</code> , <code>GetPdg()</code> , <code>GetArmID()</code> , <code>GetPlaneID()</code> , <code>GetDirectionID()</code>

Appendix B. Detector identifiers (`DetectorId`)

Values of the enumeration `DetectorId` (`base/BmnDetectorList.h`) used in hits and MC tracks:

Value	Name	Detector	Value	Name	Detector
0	<code>kREF</code>	Reference plane	14	<code>kSiMD</code>	Silicon Multiplicity Detector
1	<code>kBD</code>	Barrel Detector	15	<code>kSiBT</code>	Silicon Beam Tracker
2	<code>kGEM</code>	GEM detectors	16	<code>kSiProf</code>	Silicon beam profilometer
3	<code>kTOF1</code>	TOF-400	17	<code>kARMTRIG</code>	SRC arm triggers
4	<code>kDCH</code>	Drift chambers	18	<code>kBC</code>	Beam counters
5	<code>kTOF</code>	TOF-700 (old code)	19	<code>kSCWALL</code>	Scintillation Wall
6	<code>kTOF701</code>	TOF-700 (since 2025)	20	<code>kHODO</code>	Forward hodoscope
7	<code>kZDC</code>	Zero Degree Calorimeter	21	<code>kFHCAL</code>	Forward Hadron Calorimeter
8	<code>kVSP</code>	Vertex Silicon Plane	22	<code>kNDET</code>	Neutron detector
9	<code>kMWPC</code>	MWPC	23	<code>kHGND</code>	Full-scale neutron detector
10	<code>kECAL</code>	Electromagnetic calorimeter	24	<code>kSSD</code>	–
11	<code>kCSC</code>	Cathode Strip Chambers	25	<code>kVACWALL</code>	Vacuum wall
12	<code>kSILICON</code>	Silicon detectors (FSD)	26	<code>kMSC</code>	MSC data
13	<code>kFD</code>	Fragment Detector			

Appendix C. Differences between simulated and experimental DSTs

The structure of the DST is the same; the table summarises what differs in availability or content.

Item	Simulated DST	Experimental DST
BmnMCInfo., BmnGlobalTrackMatch, StsTrackMatch, NdetSurfacePoint	Present	Absent
BmnSiBTHit, BmnBeamTrack, CombineVertex	Absent (no beam tracking)	Present
Extra objects in the file	FairBaseParSet, BmnFieldPar, FairGeoPassivePar	–
DstRunHeader: run number, times	Run 0; times not meaningful	From the Condition Database
DstRunHeader: beam, target, energy	From the simulation metadata	From the Condition Database
DstEventHeader: run and event IDs	MC run ID, MC event number (from 1)	DAQ run and event numbers
DstEventHeader: event time	MC event time of the generator	Absolute time of the event (UTC)
DstEventHeader: impact parameter	From the generator	0
BmnTrigInfo.	BD digits and FD MC points only	Amplitudes, waveforms, trigger input signals, T0
Hits: GetRefIndex(), links, GetType()	MC point index, MC links, true/fake flag	–1, no links, –1
StsHit coordinates	MC corrections (y shift, smearing)	Alignment and Lorentz shift corrections
StsHit::GetRefIndex()	MC point index	–(original hit index) – 1
BmnGlobalTrack::GetRefIndex(), CbmStsTrack::GetTrkID()	Matched MC track	Undefined – do not use
TOF hit time	Smeared MC time	Measured time relative to T0; depends on the TOF calibration

Appendix D. Fields that are not filled or need care

The following fields exist in the format but are not filled in the current reconstruction, contain undefined values, or need special care.

Field	Status	Recommendation
DstEventHeader::GetTriggerType()	Always 1 (default)	Use the trigger input signals of BmnTrigInfo.
DstEventHeader Z2in/Z2out/ADC fields	–100 when the BC digit branches are missing in the input	Use BmnTrigInfo. amplitudes
BmnGlobalTrack::GetPDG()	Undefined value	Do not use; identify particles with m^2 , dE/dx
BmnGlobalTrack::IsPrimary()	Always true	Use CbmStsTrack::FromVertex() and DCA cuts
BmnGlobalTrack::GetRefIndex()	Undefined in experimental DSTs	Use only for simulated data
BmnGlobalTrack::GetDedx(), GetQ()	Filled only for period 8	–
BmnGlobalTrack CSC hit indices	Always –1	Match CSC hits in the analysis if needed
BmnGlobalTrack ScWall matching	Uses a fixed ScWall position ($z = 741.5$ cm, Run 8 setup)	Match with ScWallEvent in the analysis if needed
BmnGlobalTrack::GetFlag(), GetB(), hit list	Not filled	Use the inner track in StsVector
CbmStsTrack::GetTrkID()	Undefined in experimental DSTs	Use only for simulated data
PrimaryVertex. track indices	Empty	Use MpdVertex. indices
CombineVertex χ^2 and covariance	Copied from the input vertex (not updated)	Use for x, y only
TOF β , m^2 in experimental DSTs	Depend on the TOF time calibration applied in the reconstruction	Check the TOF time distribution of the production

Field	Status	Recommendation
Layer getters of VSP, Silicon, GEM and CSC hits	Taken from the clusters via <code>FairRootManager</code> ; default values when the DST tree is read directly	Use them in FairRoot tasks, or read the cluster branches
<code>StsHit::GetTimeStamp()</code>	Cluster width asymmetry, not a time	–
TOF hit cluster indices	Exp: digit indices; Sim: undefined	–
<code>EventCentrality</code>	One calibration file (set in <code>run_reco_bmn.C</code>) is used for all run periods	Use only with a calibration for the analysed system
<code>DstRunHeader</code> start and finish times	Moscow local time stored as UTC; not meaningful for MC	Take into account when comparing with event time stamps (UTC)

Appendix E. Data volume per event

The tables give the size of the DST branches per event as stored in the file, sorted by the size. The values are typical for the Run 9 setup (Xe + Sn at 3 AGeV); they depend on the collision system, the beam energy, the detector configuration and the running conditions.

E.1 Simulated data

Branch	Size [kB/event]	Share
<code>BmnMCInfo.</code>	36.2	20.0 %
<code>StsHit</code>	33.0	18.3 %
<code>BmnGemStripHit</code>	26.1	14.4 %
<code>BmnGemLowerCluster</code>	16.8	9.3 %
<code>BmnGemUpperCluster</code>	15.8	8.7 %
<code>BmnSiliconHit</code>	6.5	3.6 %
<code>StsVector</code>	6.1	3.4 %
<code>BmnGlobalTrack</code>	6.0	3.3 %
<code>BmnSiliconLowerCluster</code>	5.1	2.8 %
<code>BmnSiliconUpperCluster</code>	4.9	2.7 %
<code>BmnCSCUpperCluster</code>	3.8	2.1 %
<code>BmnCSCLowerCluster</code>	3.8	2.1 %
<code>BmnCSCHit</code>	3.6	2.0 %
<code>FHCalEvent</code>	2.8	1.6 %
<code>BmnVSPHit</code>	2.1	1.1 %
<code>BmnTof700Hit</code>	1.5	0.8 %
<code>ScWallEvent</code>	1.5	0.8 %
<code>BmnVSPUpperCluster</code>	1.2	0.7 %
<code>BmnVSPLowerCluster</code>	1.2	0.7 %
<code>BmnTof400Hit</code>	0.9	0.5 %
<code>BmnGlobalTrackMatch</code>	0.5	0.3 %
<code>BmnTrigInfo.</code>	0.5	0.3 %
<code>HodoEvent</code>	0.2	0.1 %
<code>StsTrackMatch</code>	0.2	0.1 %
<code>MpdVertex.</code>	0.1	0.1 %
<code>PrimaryVertex.</code>	0.1	< 0.1 %

Branch	Size [kB/event]	Share
DstEventHeader.	0.1	< 0.1 %
NdetSurfacePoint	0.1	< 0.1 %
NdetEvent	0.1	< 0.1 %
EventCentrality	< 0.1	< 0.1 %
Total	180.7	100 %

E.2 Experimental data

Branch	Size [kB/event]	Share
BmnSiliconUpperCluster	217.1	26.0 %
StsHit	197.2	23.7 %
BmnGemStripHit	83.3	10.0 %
BmnGemUpperCluster	71.3	8.6 %
BmnSiliconHit	54.5	6.5 %
BmnCSCHit	45.0	5.4 %
BmnGemLowerCluster	38.5	4.6 %
BmnCSCUpperCluster	33.2	4.0 %
BmnCSCLowerCluster	29.9	3.6 %
StsVector	21.4	2.6 %
BmnGlobalTrack	21.0	2.5 %
BmnSiliconLowerCluster	8.6	1.0 %
BmnTof700Hit	2.2	0.3 %
BmnTrigInfo.	1.9	0.2 %
BmnVSPHit	1.9	0.2 %
ScWallEvent	1.2	0.1 %
FHCalEvent	1.1	0.1 %
BmnTof400Hit	1.0	0.1 %
BmnVSPUpperCluster	0.8	0.1 %
BmnVSPLowerCluster	0.7	0.1 %
BmnSiBTHit	0.5	0.1 %
BmnBeamTrack	0.4	< 0.1 %
HodoEvent	0.3	< 0.1 %
CombineVertex	0.2	< 0.1 %
MpdVertex.	0.1	< 0.1 %
DstEventHeader.	0.1	< 0.1 %
PrimaryVertex.	0.1	< 0.1 %
NdetEvent	0.1	< 0.1 %
EventCentrality	0.1	< 0.1 %
Total	833.5	100 %