# Table of Contents

**Detector Description and Geometry Offline Guide....1**
Contacts....1
DD to DD4Hep Migration....1
Introduction....1
Geometry in CMSSW....2
Building the Geometry from database sources....2
Building the Geometry directly from XML files....3
Adding New Geometry Scenario....3
Adding a new XML file....4
How to load the conditions objects from XML into the database....4
How to validate conditions and XML match....4
DDD and XML versus validation....4
XML to DB round-trip DDD validation....4
Summary of CMS Policy for Geometry XML Files....4
Related links: Detector Description and Geometry....5
Related Detector Description Documentation....5
Related Geometry Documentation....5
Review Status....5

# Detector Description and Geometry Offline Guide

Complete: 

## Contacts

- **Conveners**: Ianna Osborne (FNAL)
- **Hypernews forum**: https://hypernews.cern.ch/HyperNews/CMS/get/geometry.html

## DD to DD4Hep Migration

Please, see DD to DD4Hep Migration guide here.

## Introduction

Reconstruction and simulation software modules have different requirements of the Geometry model in CMS Software. Simulation (Geant4) requires all parts to be defined in terms of size, shape, material type and position. The Reconstruction geometry only cares about reconstructing the event from information provided by the detector. Reconstruction requires those parts of the detector that are needed to define hits and tracks. It requires the shapes, sizes and position of these parts as well as a conversion from local to global coordinates of the part in the global CMS coordinate system.

The Geometry subsystem provides the reconstruction geometry model for the CMS Software project. It provides sensitive detector identification numbers as well as extracts from the Detector Description model the relevant coordinate transformations and shape parameters. It is also the repository for the XML files that contain the information needed by the DetectorDescription subsystem to build the in-memory detector description model.

These sets of XML files provide a scenario for the simulation software. For example with the forward detectors it is the Extended scenario; without the forward detectors is Ideal scenario (for historical reasons). The in-memory model of the detector description is stored using the CMS conditions database system. The "master" information from the XML files is loaded into the conditions database as one binary large object (blob) which is a single XML file generated from the component XML files of a scenario. Processing this blob is faster than processing the sets of individual files.

The reconstruction geometry is available as objects from the conditions database as well. These objects are smaller than the full geometry and are the minimum required to run reconstruction.

The DD in-memory model is accessed by the SimG4Core/Geometry package to build the Geant4 geometry to be used in simulating the detector.

The Detector Description ( original documentation page can be found at DDD ) is a software model that represents the description of the 'ideal' CMS detector. Ideal is used to signify that it does not include the alignment of the detector. It is the best estimate of the real detectors' shapes as constructed. The model is implemented as a directed acyclic graph with LogicalParts as nodes and PosParts as the edges. This provides a hierarchical representation of the detector with parts positioned inside of other parts in a parent-child relationship. This is used to build the Geometry for both reconstruction and Simulation within the CMS Software framework.

The XML is considered the master source of information to build the in-memory model. Tools are provided to store the XML into persistent objects for retrieval via the conditions and calibration framework. Although the database is considered a derivative of the XML, it is can have versions and thus is more flexible than the

XML files which are tied to a particular release. One set of loaded objects be used across multiple releases. In theory it means that the XML can change until such a time as a new approved set of payloads or one payload is required to be changed for production or reconstruction.

Some definitions:

- **DDL** Detector Description Language is an XML based language description with the schema residing in DetectorDescription/Schema.

# Geometry in CMSSW

## Building the Geometry from database sources.

The default geometry for simulation and reconstruction in CMSSW is from the conditions database. The cmsDriver command also defaults to the database objects.

A good source of examples for using the cmsDriver command is the integration build testing (http://cmssdt.cern.ch/dev/cgi-bin/showIB.py). From there one can click on PyRelVals and see the results of the release validation as well as the cmsDriver command used for the tests. For example:

```
cmsDriver.py H200ChargedTaus_Tauola_7TeV_cfi.py -s GEN,SIM,DIGI,L1,DIGI2RAW,HLT:GRun,RAW2DIGI,L1R
 -n 10 --geometry DB --conditions auto:startup --relval 9000,100 --datatier 'GEN-SIM-DIGI-RAW-HLT
 --eventcontent FEVTDEBUGHLT --fileout file:raw.root
```

The critical option for geometry is --geometry DB. By putting --geometry Extended one uses the Extended XML configuration file. (config used which refers to actual scenario file. In order to use **any other scenario** for simulation, a customization is necessary.

**Tech note:** This is because the decision was made to keep all payloads in the global tag. Some of these are the same type of object, and in the same record (FileBlob in GeometryFileRcd). Therefore one must replace a "label" attribute in the geometry reading from the database. Here is an example of changing the geometry label usable by cmsDriver: --geometry DB:Ideal.

There are a number of standard configuration files in Configuration/StandardSequences/python that are used for building the CMSSW Geometry from database sources via the EventSetup and its use in the Conditions framework. These are just a few examples:

| **Sampling of configuration files.** | |
|---|---|
| **Purpose** | **Master configuration to use** |
| Detector description to be used in standard CMSSW job | Configuration/StandardSequences/python/GeometryDB_cff.py |
| Detector description to be used in standard CMSSW that require only the RECO geometry | Configuration/StandardSequences/python/GeometryRecoDB_cff.py |
| Detector description to be used in standard CMSSW that require only the SIM geometry | Configuration/StandardSequences/python/GeometrySimDB_cff.py |

Example: If you would like to run a job in which you need access to both reconstruction and simulation geometry and you are not making a customize fragment for cmsDriver, then you have to add these lines in your configuration file:

```
process.load("Configuration.StandardSequences.GeometryDB_cff")
process.load("Configuration.StandardSequences.FrontierConditions_GlobalTag_cff")
from Configuration.AlCa.autoCond import autoCond
process.GlobalTag.globaltag = autoCond['mc']
```

This will pick up a default GlobalTag for MC. If you would like to use a specific GlobalTag, please, specify a YOUR_PREFERRED_GLOBAL_TAG. Where YOUR_PREFERRED_GLOBAL_TAG is found on the Valid Global Tags by Release section of the Frontier Conditions guide.

## Building the Geometry directly from XML files

There are a number of standard configuration files in Configuration/StandardSequences/python to be used for defining the Geometry of CMSSW as a whole using XML files.

| Sampling of configuration files. | |
|---|---|
| **Purpose** | **Master configuration to use** |
| Detector description to be used in standard CMSSW job w/o forward detector | Configuration/StandardSequences/python/GeometryIdeal_cff.py |
| Detector description to be used in standard CMSSW job with forward detector | Configuration/StandardSequences/python/GeometryExtended_cff.py |

These are the only officially supported geometry configurations. If you would like to create your own customized configuration please start by reading the section for Developers and then try to contact people with your questions at the geometry hypernews hn-cms-geometry@cernNOSPAMPLEASE.ch.

Example: If you would like to run a job using the Ideal Geometry add this line to your configuration file:

```
process.load("Configuration.StandardSequences.GeometryIdeal_cff")
```

## Adding New Geometry Scenario

In order to integrate a new scenario with CMS configuration builder, please, follow the naming conventions when adding it to CMSSW.

Usually, a new scenario is based on a current **[BaseLabel1]** scenario and it is labeled as **[SpecificLabel2]**. For example, an **Extended** geometry scenario can be taken as a base one and it has a **X0Min** label.

1. Describe it in XML

2. Add a scenario in Geometry/CMSCommonData/python/cms**[BaseLabel1]**Geometry**[SpecificLabel2]**XML_cfi.py

3. Add a configuration for the scenario in Configuration/Geometry/python/Geometry**[BaseLabel1][SpecificLabel2]**_cff.py

and its Reco:

Geometry**[BaseLabel1][SpecificLabel2]**Reco_cff.py

4. Define a label in Configuration/StandardSequences/python/GeometryConf.py

'**[BaseLabel1][SpecificLabel2]**' : '**[BaseLabel1][SpecificLabel2]**, **[BaseLabel1][SpecificLabel2]**Reco'

*Note, for 53x full file names should be defined due to a bug in cmsConfigBuilder.*

This **[BaseLabel1][SpecificLabel2]** label can be used to define geometry scenario for a cmsDriver -g command line option.

### Adding a new XML file

All newly created files should follow the convention:

```
base/filename/physical_version_key/implementation_version_key/filename.xml
```

Eg: Geometry/GEMGeometryBuilder/data /GEMSpecs /2019 /v1/GEMSpecs.xml

Then when someone makes an improvement to something existing (that is already in use and the existing versions need to be preserved), they make a v2.

Conversely when someone is making a first implementation of an upgrade, they make a new physical_version_key eg, if the GEM suddenly decides to have a new upgrade project in 2023:

Geometry/GEMGeometryBuilder/data /GEMSpecs/2023/v1/GEMSpecs.xml

## How to load the conditions objects from XML into the database.

For Geometry maintenance you will find more detailed instruction on the new data base approach and how to load all objects from XML to the conditions database here. Database Geometry.

## How to validate conditions and XML match

### DDD and XML versus validation

- DDD versus DataBase Validation

### XML to DB round-trip DDD validation

- XML to Database DDD validation

## Summary of CMS Policy for Geometry XML Files

- When geometry is frozen before a new run period, it is entered into the Conditions DB as a set of payloads. The geometry XML files for the frozen geometry are not allowed to be changed. If a new version is needed, a new version directory must be created as described above in the "Adding a new XML file" section.
- For a private production, new XML files can be created along with an alternative DB record.
- For Phase 2, there is a lengthy period of development when XML files are changed and no corresponding DB payloads exist.

# Related links: Detector Description and Geometry

## Related Detector Description Documentation

- Upgrade Geometry Scenarios
- DevelopersGuide - how to develop/maintain Geometry subsystems XML
- PerfectGeometryAnalyzer parameters explained - how to extract information from the detector description
- SpecPars definition
- DDD XML -> HTML Tool
- DDCompareCPV tool to compare two in-memory DDCompactView graphs

## Related Geometry Documentation

- Database Geometry What it means to load a database from XML files.
- Tracker Material budget, both pixel and strips.
- TEC (Tracker End Cap) geometry
    - TEC Geometry Documentation
    - TEC Materials
    - CMS.TECSandBox
- Scenarios with modified material budget in the tracker
- TrackerMaterialBudgetplots : instructions to produce plots about material budget distribution in the tracker.
- http://www.gphysics.net/geant4/geant4-physical-volume.html

# Review Status

| Reviewer/Editor and Date (copy from screen) | Comments |
|---|---|
| MichaelCase - 27 Jan 2006 | page author |
| JennyWilliams - 02 Feb 2007 | editing to include in SWGuide |

Responsible: IannaOsborne
Last reviewed by: *Never reviewed*