G4PhotonEvaporation.cc

```
G4Fragment*
G4PhotonEvaporation::GenerateGamma(G4Fragment* nucleus)
{
  if(!isInitialised) { Initialise(); }
  G4Fragment* result = nullptr;
  G4double eexc = nucleus->GetExcitationEnergy();
  if(eexc <= Tolerance) { return result; }

  InitialiseLevelManager(nucleus->GetZ_asInt(), nucleus->GetA_asInt());

  G4double time = nucleus->GetCreationTime();

  G4double efinal = 0.0;
  G4double ratio  = 0.0;
  vShellNumber    = -1;
  G4int  JP1      = 0;
  G4int  JP2      = 0;
  G4int  multiP   = 0;
  G4bool isGamma  = true;
  G4bool isDiscrete = false;

  const G4NucLevel* level = nullptr;
  size_t ntrans = 0;

  if(fVerbose > 2) {
    G4cout << "GenerateGamma: " << " Eex= " << eexc
           << " Eexmax= " << fLevelEnergyMax << G4endl;
  }
```

```
inline void
G4PhotonEvaporation::InitialiseLevelManager(G4int Z, G4int A)
{
  if(Z != theZ || A != theA) {
    theZ = Z;
    theA = A;
    fIndex = 0;
    fLevelManager = fNuclearLevelData->GetLevelManager(theZ, theA);
    fLevelEnergyMax = fLevelManager ? fLevelManager->MaxLevelEnergy() : 0.0;
  }
}
```

G4NuclearLevelData.cc

```
const G4LevelManager*
G4NuclearLevelData::GetLevelManager(G4int Z, G4int A)
{
  if(Z < 1 || Z >= ZMAX || A < AMIN[Z] || A > AMAX[Z]) { return nullptr; }
  const G4int idx = A - AMIN[Z];
  if( !(fLevelManagerFlags[Z])[idx] ) {
    G4AutoLock l(&nuclearLevelDataMutex);
    if( !(fLevelManagerFlags[Z])[idx] ) {
      (fLevelManagers[Z])[idx] = fLevelReader->CreateLevelManager(Z, A);
      (fLevelManagerFlags[Z])[idx] = true;
    }
    l.unlock();
  }
  return (fLevelManagers[Z])[idx];
}
```

G4LevelReader.cc

```
const G4LevelManager*
G4LevelReader::CreateLevelManager(G4int Z, G4int A)
{
  std::ostringstream ss;
  ss << fDirectory << "/z" << Z << ".a" << A;
  std::ifstream infile(ss.str(), std::ios::in);

  return LevelManager(Z, A, 0, infile);
}
```

This sets the location and name of the file for the nucleus undergoing photon evaporation

```
const G4LevelManager*
G4LevelReader::LevelManager(G4int Z, G4int A, G4int nlev,
                            std::ifstream& infile)
{
  // file is not opened
  if (!infile.is_open()) {
    if(Z < 6) {
      G4ExceptionDescription ed;
      ed << " for Z= " << Z << " A= " << A
         << " is not opened!";
      G4Exception("G4LevelReader::LevelManager(..)","had014",
                  FatalException, ed, "");
    }
    return nullptr;
  }
  if (fVerbose > 1) {
    G4cout << "G4LevelReader: open file for Z= "
           << Z << " A= " << A <<  G4endl;
  }

  G4bool allLevels = fParam->StoreICLevelData();

  G4int nlevels = (0 == nlev) ? fLevelMax : nlev;
  if(fVerbose > 1) {
    G4cout << "## New isotope Z= " << Z << "  A= " << A;
    if(nlevels < fLevelMax) { G4cout << " Nlevels= " << nlevels; }
    G4cout << G4endl;
  }
  if(nlevels > fLevelMax) {
    fLevelMax = nlevels;
    vEnergy.resize(fLevelMax,0.0);
    vSpin.resize(fLevelMax,0);
    vLevel.resize(fLevelMax,nullptr);
  }
```

It seems from this that because 0 is hardcoded for “nlev”, the if statement boxed in red will never be true when neutron capture is engaged

```
G4LevelReader::G4LevelReader(G4NuclearLevelData* ptr)
  : fData(ptr),fVerbose(1),fLevelMax(632),fTransMax(145)
```