Help

```
#include "lmm1d_exoi.h"
#include "pnl/pnl_basis.h"
#include "math/mc_lmm_glassermanzhao.h"
#include "enums.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2011+2) //The "#els
static int CHK_OPT(MC_LongstaffSchwartz_CallableRangeAccrual)(void *Opt, void *M
{
  return NONACTIVE;
}
int CALC(MC_LongstaffSchwartz_CallableRangeAccrual)(void *Opt, void *Mod, Pricin
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int MC_CIF_LongstaffSchwartz(double l0, double sigma_const, int nb_factor
{
  Volatility *ptVol;
  Libor *ptLib;
  int init_mc;
  int Nbr_Maturities;
  char *CouponFlag = "CallableRangeAccrual";
  PnlVect *ContractParams = pnl_vect_create(3);

  LET(ContractParams, 0) = FixedRate;
  LET(ContractParams, 1) = LowerRangeBound;
  LET(ContractParams, 2) = UpperRangeBound;

  Nbr_Maturities = pnl_iround(last_payement_date / tenor);

  mallocLibor(&ptLib , Nbr_Maturities, tenor, l0);
  mallocVolatility(&ptVol , nb_factors, sigma_const);

  init_mc = pnl_rand_init(generator, nb_factors, NbrMCsimulation);
  if (init_mc != OK) return init_mc;

  MC_ExoticProduct_LongstaffSchwartz(CouponFlag, ContractParams, swaption_price,
```

```
  freeLibor(&ptLib);
  freeVolatility(&ptVol);
  pnl_vect_free(&ContractParams);

  return init_mc;
}

int CALC(MC_LongstaffSchwartz_CallableRangeAccrual)(void *Opt, void *Mod, Pricin
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;

  return MC_CIF_LongstaffSchwartz(ptMod->l0.Val.V_PDOUBLE,
                                  ptMod->Sigma.Val.V_PDOUBLE,
                                  ptMod->NbFactors.Val.V_ENUM.value,
                                  ptOpt->LastPaymentDate.Val.V_DATE - ptMod->T.V
                                  ptOpt->FirstExerciseDate.Val.V_DATE - ptMod->T
                                  ptOpt->Nominal.Val.V_PDOUBLE,
                                  ptOpt->FixedRate.Val.V_PDOUBLE,
                                  ptOpt->LowerRangeBound.Val.V_PDOUBLE,
                                  ptOpt->UpperRangeBound.Val.V_PDOUBLE,
                                  ptOpt->ResetPeriod.Val.V_DATE,
                                  Met->Par[0].Val.V_LONG,
                                  Met->Par[1].Val.V_ENUM.value,
                                  Met->Par[2].Val.V_ENUM.value,
                                  Met->Par[3].Val.V_INT,
                                  Met->Par[4].Val.V_INT,
                                  Met->Par[5].Val.V_ENUM.value,
                                  &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(MC_LongstaffSchwartz_CallableRangeAccrual)(void *Opt, void *M
{
  if ((strcmp(((Option *)Opt)->Name, "CallableRangeAccrual") == 0))
    return OK;
  else
    return WRONG;
}

#endif //PremiaCurrentVersion
```

```
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->init = 1;

      Met->Par[0].Val.V_LONG = 50000;
      Met->Par[1].Val.V_ENUM.value = 0;
      Met->Par[1].Val.V_ENUM.members = &PremiaEnumRNGs;
      Met->Par[2].Val.V_ENUM.value = 0;
      Met->Par[2].Val.V_ENUM.members = &PremiaEnumBasis;
      Met->Par[3].Val.V_INT = 10;
      Met->Par[4].Val.V_INT = 1;
      Met->Par[5].Val.V_ENUM.value = 0;
      Met->Par[5].Val.V_ENUM.members = &PremiaEnumAfd;

    }

  return OK;
}


PricingMethod MET(MC_LongstaffSchwartz_CallableRangeAccrual) =
{
  "MC_LongstaffSchwartz_Callable_Range_Accrual",
  {
    {"N Simulation", LONG, {100}, ALLOW},
    {"RandomGenerator", ENUM, {100}, ALLOW},
    {"Basis", ENUM, {100}, ALLOW},
    {"Dimension Approximation", INT, {100}, ALLOW},
    {"Nbr discretisation step per periode", INT, {100}, ALLOW},
    {"Martingale Measure", ENUM, {100}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(MC_LongstaffSchwartz_CallableRangeAccrual),
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(MC_LongstaffSchwartz_CallableRangeAccrual),
  CHK_ok,
  MET(Init)
```

```
};
```