Fx Swap

- 인터페이스 정의서 작성(Wrapper 클래스의 멤버변수 기준)
  <FX Swap>

I/O	번호	변수명	Type	Size	설명	Sample Data
In	1	asOfDate	Date	1	평가기준일	20240728
	2	settleDate	Date	1	상품 인도일	20240730
	3	isRCVCcyForeign	Bool	1	(Far Leg기준)외화 수취 여부	1(or 0)
	4	rcvNearAmt	Real	1	Near Leg의 수취금액	10000
	5	rcvNearDate	Date	1	Near Leg의 수취일	20240730
	6	payNearAmt	Real	1	Near Leg의 지급금액	14000000
	7	payNearDate	Date	1	Near Leg의 지급일	20240730
	8	rcvFarAmt	Real	1	Far Leg의 수취금액	14500000
	9	rcvFarDate	Date	1	Far Leg의 수취일	20240830
	10	payFarAmt	Real	1	Far Leg의 지급금액	10000
	11	payFarDate	Date	1	Far Leg의 지급일	20240830
	12	rcvCCYCurveDates	vector<Date>&	TBD	테너별 기준일	20240730
	13	rcvCCYCurveYields	vector<Rate>&	TBD	테너별 Zero rate	0.032157
	14	rcvCCYCurveDayCounter	DayCounter&	1	날짜 관행	Actual365Fixed()
	15	rcvCCYCurveInterpolator	<<template>>	1	Interpolation 방법	Linear()
	16	rcvCCYCurveCompounding	Compounding	1	복리계산 방법	Continuous
	17	rcvCCYCurveFrequency	Frequency	1	복리계산 주기	Annual
	18	payCCYCurveDates	vector<Date>&	TBD	테너별 기준일	20240730
	19	payCCYCurveYields	vector<Rate>&	TBD	테너별 Zero rate	0.032157
	20	payCCYCurveDayCounter	DayCounter&	1	날짜 관행	Actual365Fixed()

	21	payCCYCurveInterpolator	<<template>>	1	Interpolation 방법	Linear()
	22	payCCYCurveCompounding	Compounding	1	복리계산 방법	Continuous
	23	payCCYCurveFrequency	Frequency	1	복리계산 주기	Annual
	24	spotFXRate	Real	1	settle시점 spot 환율	1425
Out	25	NPV	Real	1	공정가치	0.945875

※ 커브정보 배열 설명

I/O	번호	변수명	Type	Size	설명	Sample Data
In	1	dayCounter	DayCounter&	1	날짜 관행	Actual365Fixed()
	2	interpolator	<<template>>	1	Interpolation 방법	Linear()
	3	compounding	Compounding	1	복리계산 방법	Continuous
	4	frequency	Frequency	1	복리계산 주기	Annual

논리설계

FX Swap NPV(Foreign ccy short 기준)

$$NPV_{FX_{Swap}} = PV_{RCV} - PV_{PAY} \times FX_{Spot}$$

$$PV_{RCV} = PV_{RCV\_Near} + PV_{RCV\_Far}$$

$$= RCV\_Near\_Amt \times DF_{RCV\_Near} + RCV\_Far\_Amt \times DF_{RCV\_Far}$$

$$PV_{PAY} = PV_{PAY\_Near} + PV_{PAY\_Far}$$

$$= PAY\_Near\_Amt \times DF_{PAY\_Near} + PAY\_Far\_Amt \times DF_{PAY\_Far}$$

- Pricer 함수 로직 Test Code(FX Swap):

```
Date asOfDate = Settings::instance().evaluationDate();
 Date settleDate(2, Jan, 2024);
 bool isRCVCcyForeign = true;
 Real rcvNearAmt = 10000000.0 * 1288.0;
 Date rcvNearDate(Date(2, Jan, 2024) + Period(1, TimeUnit::Days));
 Real payNearAmt = 10000000.0;
 Date payNearDate(Date(2, Jan, 2024) + Period(1, TimeUnit::Days));
 Real rcvFarAmt = 10000000.0; //
```

```
    Date rcvFarDate(2, Apr, 2024);
    Real payFarAmt = rcvFarAmt * 1281.25;
    Date payFarDate(2, Apr, 2024);
    // Define RCV Currency Zero Curve
    std::vector<Date> rcvCCYCurveDates;
    rcvCCYCurveDates.emplace_back(asOfDate);
    rcvCCYCurveDates.emplace_back(45289);
    rcvCCYCurveDates.emplace_back(45302);
    rcvCCYCurveDates.emplace_back(45309);
    rcvCCYCurveDates.emplace_back(45328);
    rcvCCYCurveDates.emplace_back(45357);
    rcvCCYCurveDates.emplace_back(45386);
    rcvCCYCurveDates.emplace_back(45418);
    rcvCCYCurveDates.emplace_back(45448);
    rcvCCYCurveDates.emplace_back(45478);
    rcvCCYCurveDates.emplace_back(45510);
    rcvCCYCurveDates.emplace_back(45540);
    rcvCCYCurveDates.emplace_back(45569);
    rcvCCYCurveDates.emplace_back(45602);
    rcvCCYCurveDates.emplace_back(45630);
    rcvCCYCurveDates.emplace_back(45663);
    rcvCCYCurveDates.emplace_back(45845);
    rcvCCYCurveDates.emplace_back(46028);
    rcvCCYCurveDates.emplace_back(46393);
    rcvCCYCurveDates.emplace_back(46758);
    rcvCCYCurveDates.emplace_back(47122);
    rcvCCYCurveDates.emplace_back(47487);
    rcvCCYCurveDates.emplace_back(47854);
    rcvCCYCurveDates.emplace_back(48219);
    rcvCCYCurveDates.emplace_back(48585);
    rcvCCYCurveDates.emplace_back(48949);
    rcvCCYCurveDates.emplace_back(49313);
    rcvCCYCurveDates.emplace_back(49678);
    rcvCCYCurveDates.emplace_back(50775);
    rcvCCYCurveDates.emplace_back(52602);
    rcvCCYCurveDates.emplace_back(54429);
    std::vector<Rate> rcvCCYYields;
    rcvCCYYields.push_back(0.054239025);
    rcvCCYYields.push_back(0.054239025);
    rcvCCYYields.push_back(0.054202933);
    rcvCCYYields.push_back(0.054184962);
    rcvCCYYields.push_back(0.054153041);
    rcvCCYYields.push_back(0.05400165);
    rcvCCYYields.push_back(0.053706658);
    rcvCCYYields.push_back(0.05322162);
    rcvCCYYields.push_back(0.052414643);
    rcvCCYYields.push_back(0.051674617);
    rcvCCYYields.push_back(0.050941667);
    rcvCCYYields.push_back(0.05017059);
    rcvCCYYields.push_back(0.04949976);
    rcvCCYYields.push_back(0.048732315);
    rcvCCYYields.push_back(0.048077527);
    rcvCCYYields.push_back(0.047319517);
    rcvCCYYields.push_back(0.043503721);
    rcvCCYYields.push_back(0.040560109);
    rcvCCYYields.push_back(0.037287486);
    rcvCCYYields.push_back(0.035634644);
    rcvCCYYields.push_back(0.0347523);
```

```
    rcvCCYYields.push_back(0.034331833);
    rcvCCYYields.push_back(0.034079231);
    rcvCCYYields.push_back(0.033973531);
    rcvCCYYields.push_back(0.03396134);
    rcvCCYYields.push_back(0.033990035);
    rcvCCYYields.push_back(0.034062185);
    rcvCCYYields.push_back(0.034149918);
    rcvCCYYields.push_back(0.034371154);
    rcvCCYYields.push_back(0.034096695);
    rcvCCYYields.push_back(0.032936455);
    DayCounter rcvCCYCurveDayCounter = Actual365Fixed();
    Linear rcvCCYCurveInterpolator = Linear();
    Compounding rcvCCYCurveCompounding = Compounding::Continuous;
    Frequency rcvCCYCurveFrequency = Frequency::Annual;
    ext::shared_ptr<YieldTermStructure> rcvCCYTermstructure = ext::make_shared<ZeroCurve>(
            rcvCCYCurveDates, rcvCCYYields, rcvCCYCurveDayCounter, rcvCCYCurveInterpolator, rcvCCYCurveCompo
    RelinkableHandle<YieldTermStructure> rcvCCYCurve;
    rcvCCYCurve.linkTo(rcvCCYTermstructure);
    // Define PAY Currency Zero Curve
    std::vector<Date> payCCYCurveDates;
    payCCYCurveDates.emplace_back(asOfDate);
    payCCYCurveDates.emplace_back(45289);
    payCCYCurveDates.emplace_back(45293);
    payCCYCurveDates.emplace_back(45300);
    payCCYCurveDates.emplace_back(45324);
    payCCYCurveDates.emplace_back(45355);
    payCCYCurveDates.emplace_back(45384);
    payCCYCurveDates.emplace_back(45475);
    payCCYCurveDates.emplace_back(45567);
    payCCYCurveDates.emplace_back(45659);
    payCCYCurveDates.emplace_back(45845);
    payCCYCurveDates.emplace_back(46028);
    payCCYCurveDates.emplace_back(46210);
    payCCYCurveDates.emplace_back(46393);
    payCCYCurveDates.emplace_back(46575);
    payCCYCurveDates.emplace_back(46758);
    payCCYCurveDates.emplace_back(46940);
    payCCYCurveDates.emplace_back(47122);
    payCCYCurveDates.emplace_back(47304);
    payCCYCurveDates.emplace_back(47487);
    payCCYCurveDates.emplace_back(47669);
    payCCYCurveDates.emplace_back(47854);
    payCCYCurveDates.emplace_back(48036);
    payCCYCurveDates.emplace_back(48219);
    payCCYCurveDates.emplace_back(48402);
    payCCYCurveDates.emplace_back(48585);
    payCCYCurveDates.emplace_back(48767);
    payCCYCurveDates.emplace_back(48949);
    payCCYCurveDates.emplace_back(49678);
    payCCYCurveDates.emplace_back(50775);
    payCCYCurveDates.emplace_back(52602);
    std::vector<Rate> payCCYYields;
    payCCYYields.push_back(0.0259214021144641);
    payCCYYields.push_back(0.0259214021144641);
    payCCYYields.push_back(-0.0024271082976222);
    payCCYYields.push_back(0.0164296679449468);
    payCCYYields.push_back(0.0277753618200306);
    payCCYYields.push_back(0.0302929872912781);
```

```
    payCCYYields.push_back(0.0307983104097034);
    payCCYYields.push_back(0.0302800226737143);
    payCCYYields.push_back(0.0293460226118934);
    payCCYYields.push_back(0.0278313271506498);
    payCCYYields.push_back(0.0259502886916105);
    payCCYYields.push_back(0.0241413034217803);
    payCCYYields.push_back(0.0237130049102298);
    payCCYYields.push_back(0.0232477839136826);
    payCCYYields.push_back(0.0230428006219736);
    payCCYYields.push_back(0.0228354657849537);
    payCCYYields.push_back(0.0227229820455438);
    payCCYYields.push_back(0.0226073665198862);
    payCCYYields.push_back(0.0224085265342971);
    payCCYYields.push_back(0.0222093011644315);
    payCCYYields.push_back(0.0221414678028452);
    payCCYYields.push_back(0.0220696299836975);
    payCCYYields.push_back(0.0219152743512371);
    payCCYYields.push_back(0.0217622795432572);
    payCCYYields.push_back(0.0219121976761305);
    payCCYYields.push_back(0.022051973444772);
    payCCYYields.push_back(0.0219723002791276);
    payCCYYields.push_back(0.0218946065793662);
    payCCYYields.push_back(0.0216899204306834);
    payCCYYields.push_back(0.0213150611594707);
    payCCYYields.push_back(0.0208780265833001);
    DayCounter payCCYCurveDayCounter = Actual365Fixed();
    Linear payCCYCurveInterpolator = Linear();
    Compounding payCCYCurveCompounding = Compounding::Continuous;
    Frequency payCCYCurveFrequency = Frequency::Annual;
    ext::shared_ptr<YieldTermStructure> payCCYTermstructure = ext::make_shared<ZeroCurve>(
            payCCYCurveDates, payCCYYields, payCCYCurveDayCounter, payCCYCurveInterpolator, payCCYCurveCompo
    RelinkableHandle<YieldTermStructure> payCCYCurve;
    payCCYCurve.linkTo(payCCYTermstructure);
    Real spotFXRate = 1288.0;
    // Assign FX spot rate for Reference Currency
    Real rcvSpotFXRate = 1.0;
    Real paySpotFXRate = 1.0;
    if( isRCVCcyForeign == true )
    {
        rcvSpotFXRate = spotFXRate;
    } else {
        paySpotFXRate = spotFXRate;
    }
    // Calculate Present value of Far Leg CF
    DiscountFactor rcvSettleDF = rcvCCYCurve->discount(settleDate);
    DiscountFactor rcvPaymentDF = rcvCCYCurve->discount(rcvFarDate);
    DiscountFactor rcvForwardDF = rcvPaymentDF / rcvSettleDF;
    DiscountFactor paySettleDF = payCCYCurve->discount(settleDate);
    DiscountFactor payPaymentDF = payCCYCurve->discount(payFarDate);
    DiscountFactor payForwardDF = payPaymentDF / paySettleDF;
    Real farLegNPV = 0.0;
    farLegNPV = rcvSpotFXRate * rcvFarAmt * rcvForwardDF - paySpotFXRate * payFarAmt * payForwardDF;
    // Calculate Present value of Far Leg CF
    // The currency applied to the near leg amount is opposite to the currency applied to the far leg amount
    // (By definition of the FX Swap instrument)
    Real rcvNearSpotFXRate = paySpotFXRate;
    Real payNearSpotFXRate = rcvSpotFXRate;
    // Calculate Present value of Far Leg CF
```

```
    DiscountFactor rcvNearSettleDF = payCCYCurve->discount(settleDate);
    DiscountFactor rcvNearPaymentDF = payCCYCurve->discount(rcvNearDate);
    DiscountFactor rcvNearForwardDF = rcvNearPaymentDF / rcvNearSettleDF;
    DiscountFactor payNearSettleDF = rcvCCYCurve->discount(settleDate);
    DiscountFactor payNearPaymentDF = rcvCCYCurve->discount(payNearDate);
    DiscountFactor payNearForwardDF = payNearPaymentDF / payNearSettleDF;
    Real nearLegNPV = 0.0;
    if( rcvNearDate > settleDate )
    {
        nearLegNPV += rcvNearSpotFXRate * rcvNearAmt * rcvNearForwardDF;
    }
    if( payNearDate > settleDate )
    {
        nearLegNPV -= payNearSpotFXRate * payNearAmt * payNearForwardDF;
    }
    Real NPV = 0.0;
    NPV = nearLegNPV + farLegNPV;
    std::cout.precision(16);
    std::cout << "NPV: " << NPV << std::endl;

```

- Wrapper 클래스 개발
-