NAME

CreditDefaultSwap − Credit default swap.

SYNOPSIS

`#include <ql/instruments/creditdefaultswap.hpp>`

Inherits **Instrument**.

Public Member Functions

Constructors

CreditDefaultSwap (Protection::Side side, **Real** notional, **Rate** spread, const **Schedule** &schedule, **BusinessDayConvention** paymentConvention, const **DayCounter** &dayCounter, bool settlesAccrual=true, bool paysAtDefaultTime=true, const **Date** &protectionStart=**Date**(), const boost::shared_ptr< **Claim** > &=boost::shared_ptr< **Claim** >())
CDS quoted as running-spread only.
CreditDefaultSwap (Protection::Side side, **Real** notional, **Rate** upfront, **Rate** spread, const **Schedule** &schedule, **BusinessDayConvention** paymentConvention, const **DayCounter** &dayCounter, bool settlesAccrual=true, bool paysAtDefaultTime=true, const **Date** &protectionStart=**Date**(), const **Date** &upfrontDate=**Date**(), const boost::shared_ptr< **Claim** > &=boost::shared_ptr< **Claim** >())
CDS quoted as upfront and running spread.

Inspectors

Protection::Side **side** () const
Real notional () const
Rate runningSpread () const
boost::optional< **Rate** > **upfront** () const
bool **settlesAccrual** () const
bool **paysAtDefaultTime** () const
const **Leg** & **coupons** () const
const **Date** & **protectionStartDate** () const
The first date for which defaults will trigger the contract.
const **Date** & **protectionEndDate** () const
The last date for which defaults will trigger the contract.

Results

Rate fairUpfront () const
Rate fairSpread () const
Real couponLegBPS () const
Real upfrontBPS () const
Real couponLegNPV () const
Real defaultLegNPV () const
Real upfrontNPV () const
Rate impliedHazardRate (**Real** targetNPV, const **Handle**< **YieldTermStructure** > &discountCurve, const **DayCounter** &dayCounter, **Real** recoveryRate=0.4, **Real** accuracy=1.0e−6) const
Implied hazard rate calculation.
Rate conventionalSpread (**Real** conventionalRecovery, const **Handle**< **YieldTermStructure** > &discountCurve, const **DayCounter** &dayCounter) const
Conventional/standard upfront-to-spread conversion.

Protected Attributes

Protection::Side **side_**
Real notional_
boost::optional< **Rate** > **upfront_**
Rate runningSpread_
bool **settlesAccrual_**

bool **paysAtDefaultTime_**
boost::shared_ptr< **Claim** > **claim_**
Leg leg_
boost::shared_ptr< **CashFlow** > **upfrontPayment_**
Date protectionStart_
Rate fairUpfront_
Rate fairSpread_
Real couponLegBPS_
Real couponLegNPV_
Real upfrontBPS_
Real upfrontNPV_
Real defaultLegNPV_

Instrument interface

bool **isExpired** () const
returns whether the instrument might have value greater than zero.
void **setupArguments** (PricingEngine::arguments *) const
void **fetchResults** (const PricingEngine::results *) const
void **setupExpired** () const

Additional Inherited Members

Detailed Description

Credit default swap.

Note:
This instrument currently assumes that the issuer did not default until today's date.

Warning
if `Settings::includeReferenceDateCashFlows()` is set to `true`, payments occurring at the settlement date of the swap might be included in the NPV and therefore affect the fair-spread calculation. This might not be what you want.

Examples:
CDS.cpp.

Constructor & Destructor Documentation

CreditDefaultSwap (Protection::Side side, Real notional, Rate spread, const Schedule & schedule, BusinessDayConvention paymentConvention, const DayCounter & dayCounter, bool settlesAccrual = `true`**, bool paysAtDefaultTime =** `true`**, const Date & protectionStart = Date**`()`**, const boost::shared_ptr< Claim > & =** `boost::shared_ptr<` **Claim** `>())`
CDS quoted as running-spread only.

Parameters:
side Whether the protection is bought or sold.
notional Notional value
spread Running spread in fractional units.
schedule **Coupon** schedule.
paymentConvention Business-day convention for payment-date adjustment.
dayCounter Day-count convention for accrual.
settlesAccrual Whether or not the accrued coupon is due in the event of a default.
paysAtDefaultTime If set to true, any payments triggered by a default event are due at default time. If set to false, they are due at the end of the accrual period.
protectionStart The first date where a default event will trigger the contract.

CreditDefaultSwap (Protection::Side side, Real notional, Rate upfront, Rate spread, const Schedule & schedule, BusinessDayConvention paymentConvention, const DayCounter & dayCounter, bool settlesAccrual = `true`**, bool paysAtDefaultTime =** `true`**, const Date & protectionStart = Date**`()`**, const Date & upfrontDate = Date**`()`**, const boost::shared_ptr< Claim > & =** `boost::shared_ptr<` **Claim** `>())`
CDS quoted as upfront and running spread.

Parameters:

side Whether the protection is bought or sold.
notional Notional value
upfront Upfront in fractional units.
spread Running spread in fractional units.
schedule **Coupon** schedule.
paymentConvention Business-day convention for payment-date adjustment.
dayCounter Day-count convention for accrual.
settlesAccrual Whether or not the accrued coupon is due in the event of a default.
paysAtDefaultTime If set to true, any payments triggered by a default event are due at default time. If set to false, they are due at the end of the accrual period.
protectionStart The first date where a default event will trigger the contract.
upfrontDate **Settlement** date for the upfront payment.

Member Function Documentation

void setupArguments (PricingEngine::arguments *) const `[virtual]`

When a derived argument structure is defined for an instrument, this method should be overridden to fill it. This is mandatory in case a pricing engine is used.

Reimplemented from **Instrument**.

void fetchResults (const PricingEngine::results * r) const `[virtual]`

When a derived result structure is defined for an instrument, this method should be overridden to read from it. This is mandatory in case a pricing engine is used.

Reimplemented from **Instrument**.

Rate fairUpfront () const

Returns the upfront spread that, given the running spread and the quoted recovery rate, will make the instrument have an NPV of 0.

Rate fairSpread () const

Returns the running spread that, given the quoted recovery rate, will make the running-only CDS have an NPV of 0.

Note:

This calculation does not take any upfront into account, even if one was given.

Examples:

CDS.cpp.

Real couponLegBPS () const

Returns the variation of the fixed-leg value given a one-basis-point change in the running spread.

Rate impliedHazardRate (Real targetNPV, const Handle< YieldTermStructure > & discountCurve, const DayCounter & dayCounter, Real recoveryRate = `0.4`**, Real accuracy =** `1.0e-6`**) const**

Implied hazard rate calculation.

Note:

This method performs the calculation with the instrument characteristics. It will coincide with the ISDA calculation if your object has the standard characteristics. Notably:

- The calendar should have no bank holidays, just weekends.
- The yield curve should be LIBOR piecewise constant in fwd rates, with a discount factor of 1 on the calculation date, which coincides with the trade date.
- Convention should be Following for yield curve and contract cashflows.
- The CDS should pay accrued and mature on standard **IMM** dates, settle on trade date +1 and upfront settle on trade date +3.

Rate conventionalSpread (Real conventionalRecovery, const Handle< YieldTermStructure > & discountCurve, const DayCounter & dayCounter) const

Conventional/standard upfront-to-spread conversion. Under a standard ISDA model and a set of standardised instrument characteristics, it is the running only quoted spread that will make a CDS contract have an NPV of 0 when quoted for that running only spread. Refer to: 'ISDA Standard CDS converter specification.' May 2009.

The conventional recovery rate to apply in the calculation is as specified by ISDA, not necessarily equal to the market-quoted one. It is typically 0.4 for SeniorSec and 0.2 for subordinate.

Note:

The conversion employs a flat hazard rate. As a result, you will not recover the market quotes.

This method performs the calculation with the instrument characteristics. It will coincide with the ISDA calculation if your object has the standard characteristics. Notably:

- The calendar should have no bank holidays, just weekends.
- The yield curve should be LIBOR piecewise constant in fwd rates, with a discount factor of 1 on the calculation date, which coincides with the trade date.
- Convention should be Following for yield curve and contract cashflows.
- The CDS should pay accrued and mature on standard **IMM** dates, settle on trade date +1 and upfront settle on trade date +3.

void setupExpired () const `[protected], [virtual]`

This method must leave the instrument in a consistent state when the expiration condition is met.

Reimplemented from **Instrument**.

Author

Generated automatically by Doxygen for QuantLib from the source code.