

```
#ifndef INTRADAY_RECORD_H
#define INTRADAY_RECORD_H
```

```
/******
```

```
// s_IntradayRecord
```

```
//Special values used for s_IntradayRecord::Open
```

```
const float SINGLE_TRADE_WITH_BID_ASK = 0.0F;
```

```
const float FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE = -1.99900095e+37F;
```

```
const float LAST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE = -1.99900197e+37F;
```

```
struct s_IntradayRecord
```

```
{
```

```
    ///--- Static Members -----
```

```
    ///--- Members -----
```

```
    SCDatetimeMS DateTime;
```

```
    float Open;
```

```
    float High;
```

```
    float Low;
```

```
    float Close;
```

```
    uint32_t NumTrades;
```

```
    uint32_t TotalVolume;
```

```
    uint32_t BidVolume;
```

```
    uint32_t AskVolume;
```

```
    ///--- Methods -----
```

```
    s_IntradayRecord();
```

```
    s_IntradayRecord(int Value);
```

```
    bool operator == (const s_IntradayRecord& Record) const;
```

```
    s_IntradayRecord& operator = (const s_IntradayRecord& Record);
```

```
    s_IntradayRecord& operator += (const float Amount);
```

```
    s_IntradayRecord& operator *= (const float Multiplier);
```

```
    s_IntradayRecord& operator /= (const float Divisor);
```

```
    void AddAdjustment
```

```
        ( const float Amount
```

```
        , const bool AdjustZeroValues
```

```
        );
```

```
    bool IsEmpty() const;
```

```
    void Clear();
```

```
    void ClearOpenFlags();
```

```
    void Fix();
```

```
    bool IsSingleTradeWithBidAsk() const;
```

```
    bool IsFirstSubTradeOfUnbundledTrade() const;
```

```
    bool IsLastSubTradeOfUnbundledTrade() const;
```

```
    bool IsFirstOrLastSubTradeOfUnbundledTrade() const;
```

```
    bool IsBidAskUpdateOnly() const;
```

```
    float GetOpen() const;
```

```
    float GetHigh() const;
```

```
    float GetLow() const;
```

```
    float GetClose() const;
```

```

float GetBidPrice() const;
float GetAskPrice() const;
};

/*****
// s_Header

struct s_IntradayFileHeader
{
    static const uint32_t UNIQUE_HEADER_ID = 0x44494353; // "SCID"

    uint32_t FileTypeUniqueHeaderID; // "SCID"
    uint32_t HeaderSize;

    uint32_t RecordSize;
    uint16_t Version;

    uint16_t Unused1;

    uint32_t Unused2;

    char Reserve[36];

    s_IntradayFileHeader()
        : FileTypeUniqueHeaderID(UNIQUE_HEADER_ID)
        , HeaderSize(sizeof(s_IntradayFileHeader))
        , RecordSize(sizeof(s_IntradayRecord))
        , Version(1)
        , Unused1(0)
        , Unused2(0)
    {
        memset(Reserve, 0, sizeof(Reserve));
    }
};

/*****
// s_IntradayRecord

/*=====*/
inline s_IntradayRecord::s_IntradayRecord()
{
    Clear();
}

/*=====
This constructor is to support using this structure as an c_SCArray
element.
-----*/
inline s_IntradayRecord::s_IntradayRecord(int Value)
{
    Clear();
}

/*=====*/
inline bool s_IntradayRecord::operator ==
(const s_IntradayRecord& Record
) const
{
    return memcmp(this, &Record, sizeof(s_IntradayRecord)) == 0;
}

/*=====*/

```

```

inline s_IntradayRecord&
s_IntradayRecord::operator =
(const s_IntradayRecord& Record
)
{
    if (&Record == this)
        return *this;

    memcpy(this, &Record, sizeof(s_IntradayRecord));

    return *this;
}

/*=====*/
inline s_IntradayRecord&
s_IntradayRecord::operator += (const float Amount)
{
    if (!IsSingleTradeWithBidAsk())
        Open += Amount;

    High += Amount;
    Low += Amount;
    Close += Amount;

    return *this;
}

/*=====*/
inline s_IntradayRecord&
s_IntradayRecord::operator *= (const float Multiplier)
{
    if (!IsSingleTradeWithBidAsk())
        Open *= Multiplier;

    High *= Multiplier;
    Low *= Multiplier;
    Close *= Multiplier;

    return *this;
}

/*=====*/
inline s_IntradayRecord&
s_IntradayRecord::operator /= (const float Divisor)
{
    if (!IsSingleTradeWithBidAsk())
        Open /= Divisor;

    High /= Divisor;
    Low /= Divisor;
    Close /= Divisor;

    return *this;
}

/*=====
Adds the given Amount to the prices, so long as the prices are non-zero,
or AdjustZeroValues is true.
-----*/
inline void s_IntradayRecord::AddAdjustment
(const float Amount
, const bool AdjustZeroValues
)
{
    if (!IsSingleTradeWithBidAsk())
    {

```

```

        if (Open != 0.0f || AdjustZeroValues)
            Open += Amount;
    }

    if (High != 0.0f || AdjustZeroValues)
        High += Amount;

    if (Low != 0.0f || AdjustZeroValues)
        Low += Amount;

    if (Close != 0.0f || AdjustZeroValues)
        Close += Amount;
}

/*=====*/
inline bool s_IntradayRecord::IsEmpty() const
{
    return DateTime.IsUnset()
        && Open == 0.0f
        && High == 0.0f
        && Low == 0.0f
        && Close == 0.0f
        && NumTrades == 0
        && TotalVolume == 0;
}

/*=====*/
inline void s_IntradayRecord::Clear()
{
    DateTime.Clear();
    Open = 0.0f;
    High = 0.0f;
    Low = 0.0f;
    Close = 0.0f;
    NumTrades = 0;
    TotalVolume = 0;
    BidVolume = 0;
    AskVolume = 0;
}

/*=====*/
inline void s_IntradayRecord::ClearOpenFlags()
{
    Open = 0.0f;
}

/*=====*/
This function fixes any data that looks erroneous.
-----*/
inline void s_IntradayRecord::Fix()
{
    if (NumTrades > INT_MAX)
        NumTrades = 0;

    if (TotalVolume > INT_MAX)
        TotalVolume = 0;

    if (BidVolume > INT_MAX)
        BidVolume = 0;

    if (AskVolume > INT_MAX)
        AskVolume = 0;
}

/*=====*/

```

```

inline bool s_IntradayRecord::IsSingleTradeWithBidAsk() const
{
    return NumTrades == 1
        && (Open == 0.0
            || Open == FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE
            || Open == LAST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE
            );
}

/*=====*/
inline bool s_IntradayRecord::IsFirstSubTradeOfUnbundledTrade() const
{
    return (Open == FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE);
}

/*=====*/
inline bool s_IntradayRecord::IsLastSubTradeOfUnbundledTrade() const
{
    return (Open == LAST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE);
}

/*=====*/
inline bool s_IntradayRecord::IsFirstOrLastSubTradeOfUnbundledTrade() const
{
    return Open == FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE
        || Open == LAST_SUB_TRADE_OF_UNBUNDLED_TRADE_VALUE;
}

/*=====*/
inline bool s_IntradayRecord::IsBidAskUpdateOnly() const
{
    return NumTrades == 1
        && TotalVolume == 0
        && Open == 0.0
        && High != 0.0
        && Low != 0.0;
}

/*=====*/
inline float s_IntradayRecord::GetOpen() const
{
    if (IsSingleTradeWithBidAsk())
        return Close;

    return Open;
}

/*=====*/
inline float s_IntradayRecord::GetHigh() const
{
    if (IsSingleTradeWithBidAsk())
        return Close;

    return High;
}

/*=====*/
inline float s_IntradayRecord::GetLow() const
{
    if (IsSingleTradeWithBidAsk())
        return Close;

    return Low;
}

```

```
/*=====*/
inline float s_IntradayRecord::GetClose() const
{
    return Close;
}

/*=====*/
inline float s_IntradayRecord::GetBidPrice() const
{
    if (!IsSingleTradeWithBidAsk())
        return 0.0f;

    return Low;
}

/*=====*/
inline float s_IntradayRecord::GetAskPrice() const
{
    if (!IsSingleTradeWithBidAsk())
        return 0.0f;

    return High;
}
#endif
```