

Intraday Data File Format

- [Introduction and General Information](#)
 - [Data Structures Used](#)
 - [s_IntradayRecord Structure Member Descriptions](#)
 - [DateTime](#)
 - [Open](#)
 - [High](#)
 - [Low](#)
 - [Close](#)
 - [NumTrades](#)
 - [TotalVolume](#)
 - [BidVolume](#)
 - [AskVolume](#)
 - [How to Feed Sierra Chart Data Through Intraday Data Files](#)
-

Introduction and General Information

This page explains the format for Intraday data files used in Sierra Chart. Intraday data files are the files that contain data for Intraday charts. They simply contain only the actual price/value and volume data, nothing else. There are two purposes for this documentation. The first is so that you can read the data in other programs and the second is so that you can feed Sierra Chart data from your own source.

An Intraday chart data file first begins with a single **header** of type **s_IntradayHeader** followed by multiple **records** of type **s_IntradayRecord** for each time unit of price/value data for a symbol. The file extension is **.scid**. When connected to a streaming data feed, Intraday data files are continuously updated. Sierra Chart opens the file in shared mode when it reads or writes to the file.

If you are creating the Intraday data files yourself and writing data to them, when the file is opened as a chart in Sierra Chart, the chart will read the data in the file beginning at the Date which is calculated from the last Date in the file minus the **Days to Load** setting in **Chart >> Chart Settings**.

New data appended to the file will be read from the file at the rate set by the **Chart Update Interval** setting in **Global Settings >> General Settings**.

Data in a Sierra Chart Intraday Data file (File extension: **.scid**) is a non-text format. It can be exported to a text format. To do this, refer to [Exporting and Importing Intraday Data Files](#).

When Sierra Chart is writing received streaming data to Intraday chart data files, it does not write to them every trade or bid/ask update, it will write to an internal buffer and then this data is then written to the file every 5 seconds. The flush time can be changed through

Global Settings >> Data/Trade Service Settings >> Advanced >> Intraday File Flush Time in Milliseconds. If set to 0, then a default flush time is used.

Data Structures Used

The Intraday data file header, `s_IntradayHeader`, is 56 bytes in size.

```
struct s_IntradayHeader
{
    char FileTypeUniqueHeaderID[4]; // Set to the text string: "SCID"
    u_int32 HeaderSize; // Set to the header size in bytes.

    u_int32 RecordSize; // Set to the record size in bytes.
    u_int16 Version; // Automatically set to the current version. Currently 1.
    u_int16 Unused1; // Not used.

    u_int32 UTCStartIndex; // This should be 0.

    char Reserve[36]; // Not used.

    s_IntradayHeader()
    {
        strncpy (FileTypeUniqueHeaderID, "SCID", 4);
        HeaderSize = sizeof(s_IntradayHeader);
        RecordSize = sizeof(s_Record);
        Version = 1;
        Unused1 = 0;
        UTCStartIndex = 0;
        memset(Reserve, 0, sizeof(Reserve));
    }
};
```

The Intraday data record structure, **`s_IntradayRecord`**, is 40 bytes in size.

```

struct s_IntradayRecord
{
    static const float SINGLE_TRADE_WITH_BID_ASK;
    static const float FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE;
    static const float LAST_SUB_TRADE_OF_UNBUNDLED_TRADE;

    SCDatetimeMS DateTime;

    float Open;
    float High;
    float Low;
    float Close;

    u_int32 NumTrades;
    u_int32 TotalVolume;
    u_int32 BidVolume;
    u_int32 AskVolume;

    s_IntradayRecord()// Constructor
    {
        DateTime;
        Open = 0.0;
        High = 0.0;
        Low = 0.0;
        Close = 0.0;
        NumTrades = 0;
        TotalVolume = 0;
        BidVolume = 0;
        AskVolume = 0;
    }
};

//Definitions for the above constants in s_IntradayRecord.
const float s_IntradayRecord::SINGLE_TRADE_WITH_BID_ASK = 0.0F;
const float s_IntradayRecord::FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE = -1.99900095e+37F;
const float s_IntradayRecord::LAST_SUB_TRADE_OF_UNBUNDLED_TRADE = -1.99900197e+37F;

```

s_IntradayRecord Structure Member Descriptions

DateTime

The **DateTime** member variable is a [SCDateTimeMS](#) variable.

This is a 64-bit integer. It represents the number of microseconds since the [SCDateTime epoch](#) of December 30, 1899.

Versions prior to 2151, use a double precision floating point type variable to represent Date-Time values. For a complete explanation of the Date component of this value, refer to [Date Value](#). The date value is the integer portion of the double. The fractional portion is the Time value which is represented as a fraction of one day where 1/86400000 is 1 ms. 86400000 is the number of milliseconds in a day.

This Date and Time value is and must be in the UTC time zone.

In Sierra Chart, if the **Intraday Data Storage Time Unit** in **Global Settings >>Data/Trade Service Settings** is set to a value greater than **1 Second** like **1**

Minute and the first trade within a new minute begins at a time after the beginning of the minute, for example 9:28:22, the time value part of the **DateTime** member in the record will be set to this time.

A new record will begin if there is any trading on or after 9:29:00 and not 9:29:22. The data is aligned on time boundaries which are multiples of the **Intraday Data Storage Time Unit**.

When writing tick by tick data to an Intraday data file, where each tick is a single record in the file, and multiple ticks/trades have the same timestamp, then it is acceptable for the Date-Time of the record to be the same timestamp as prior records. What Sierra Chart itself does in this case, is increment the microsecond portion of the Date-Time to create unique timestamps for each trade within a millisecond.

Sierra Chart provides the **/ACS_Source/SCDateTime.h** file with various functions for working with these Date-Time values.

Open

The opening value of the data record stored in a 4-byte floating point format.

In the case where the data record holds 1 tick/trade of data, the Open will be equal to 0 (defined by the `SINGLE_TRADE_WITH_BID_ASK` constant).

The Open field can also contain the following special values:

`FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE`,

`LAST_SUB_TRADE_OF_UNBUNDLED_TRADE`. Refer to [Data Structures Used](#). These are used in the case of CME data with both of the [Real-Time Exchange Data Feeds Available from Sierra Chart](#) as well as the [Sierra Chart Historical Data Service](#) and indicate the first and last sub trades which are part of a larger summary trade.

High

The high value of the data record stored in a 4-byte floating point format.

In the case where the data record holds 1 tick/trade of data, the High will be equal to Ask price at the time of the trade.

Low

The low value of the data record stored in a 4-byte floating point format.

In the case where the data record holds 1 tick/trade of data, the Low will be equal to Bid price at the time of the trade.

Close

The closing value of the data record stored in a 4-byte floating point format.

In the case where the data record holds 1 tick/trade of data, the Close will be equal to the actual price of the trade.

NumTrades

The the number of trades in the data record stored in a 4-byte integer format. This value could be zero.

TotalVolume

The total volume of the data record stored in a 4-byte integer format. This value could be zero.

BidVolume

The total bid volume of the data record stored in a 4-byte integer format. This is the volume of the trades that occurred at the bid price or lower. This value could be zero.

When this is a nonzero value and the **NumTrades** is equal to 1, then this indicates that the trade occurred at the Bid price. Or another way to say this is the aggressor of the trade was on the sell side.

AskVolume

The total ask volume of the data record stored in a 4-byte integer format. This is the volume of the trades that occurred at the ask price or higher. This value could be zero.

When this is a nonzero value and the **NumTrades** is equal to 1, then this indicates that the trade occurred at the Ask price. Or another way to say this is the aggressor of the trade was on the buy side.

How to Feed Sierra Chart Data Through Intraday Data Files

This section explains in general how to feed Sierra Chart data from an external source by creating and updating Intraday data files in the format documented here.

A more modern and up-to-date method of feeding Sierra Chart data is through the [DTC Protocol](#) which Sierra Chart fully supports.

Although updating Intraday chart data files directly is a reasonable method. The data received by the Data and Trading services that Sierra Chart is integrated with, through the Sierra Chart program, also write to Intraday chart data files.

1. First, create a file with the following file name format: **[symbol].scid** . This file must be located in the **Sierra Chart Data Files Folder** which is specified through **Global Settings >> General Settings** .

If the Intraday data file you are creating is not located in this folder and you try to open it from another location with **File >> New/Open Intraday Chart** , Sierra Chart will automatically change the path to the **Data Files Folder** and create a new file. It will then be reading an empty data file.

2. The file must be opened with file share read and file share write access. This is essential.

3. Write an Intraday file header of type **s_IntradayHeader** to the beginning of the file. Once this is done, it does not need to be done again. For an understanding of the different fields of the header, see the structure definition above and the comments.
4. Write Intraday data records to the file. The structure type of these records is **s_IntradayRecord** . These records are written immediately after the header and are in sequence. The first record represents the first and earliest time and the last record represents the last and latest time. An Intraday data record, of type **s_IntradayRecord** , can represent 1 tick or any specific period of time such as 1 second, 10 seconds, 1 minute or any other period up to 1 day. For daily data which represents one day of trading, we use an [Text data format](#).
5. You will write Intraday data records to the file as data is received in real time from your data feed or to fill the file with historical data.

If a data record represents one single tick/trade, then the Intraday data record Open, High, Low, and Close can be set to the same value and you will write one record for every tick/trade.

If you want to record the Bid and Ask price for the trade, then set the Open to 0, the High to the Ask price, the Low to the Bid price, and the Close to the trade price.

If a data record represents a specific period of time, such as 10 seconds, then the Open, High, Low, and Close and Volume fields are for that 10 second period of time. You will write one record for every 10 seconds of time there is trading activity. If there is no trading activity for a particular period of time, then no record needs to be written.

6. Example: When you are writing data in real-time and the record represents a specific period of time other than one tick/trade, and your Intraday record length is 1 minute, and 10 trades have occurred, then what you will do is add a new record on the first trade, and update that record 9 additional times when those trades occur. When you initially add the record, the Open, High, Low, and Close will all be the same value. When you update it 9 additional times, first read the last record written, increase the High if necessary, decrease the Low if necessary, update the Close, increase all of the Volume fields, and write the updated record back to the same location in the file.

*Last modified Wednesday, 22nd February, 2023.