

```
#ifndef ACSIL_CUSTOM_CHART_BARS_H
#define ACSIL_CUSTOM_CHART_BARS_H
```

```
struct s_CustomChartBarInterface
{
```

```
public:
```

```
    s_CustomChartBarInterface();
```

```
    //All of the following are Input members
```

```
    ChartDataTypeEnum ChartDataType; //Indicates if this chart is based on historical Daily data or Intraday data.
```

```
    unsigned char IsDeterminingIfShouldStartNewBar; //1 = if the custom bar building function needs to indicate if should
start a new chart bar. Otherwise, it is 0.
```

```
    unsigned char IsFinalProcessingAfterNewOrCurrentBar;
```

```
    unsigned char IsInsertFileRecordsProcessing;
```

```
    s_IntradayRecord NewFileRecord;
```

```
    uint32_t CurrentBarIndex;
```

```
    int32_t ValueFormat;
```

```
    unsigned char IsNewFileRecord;
```

```
    unsigned char BarHasBeenCutAtStartOfSession;
```

```
    unsigned char IsNewChartBar;
```

```
    unsigned char IsFirstBarOfChart; //1 = When the first bar of the chart
```

```
    float TickSize;
```

```
    //The following are Output members
```

```
    unsigned char StartNewBarFlag; //Set to 1 to start a new chart bar. In this case NewFileRecord becomes part of the new
chart bar.
```

```
    unsigned char InsertNewRecord;
```

```
    s_IntradayRecord NewRecordToInsert;
```

```
    // Functions
```

```
    float& GetChartBarValue(int SubgraphIndex, int BarIndex);
```

```
    const SCDatetime& GetChartBarDateTime(int BarIndex);
```

```
    SCInputRef& GetInput(int InputIndex);
```

```
    int& GetPersistentInt(int Key);
```

```
    float& GetPersistentFloat(int Key);
```

```
    double& GetPersistentDouble(int Key);
```

```
    int64_t& GetPersistentInt64(int Key);
```

```
    SCDatetime& GetPersistentSCDateTime(int Key);
```

```
    void SetLoadingDataObjectPointer(void* p_LoadingDataObject)
```

```
    {
        m_p_LoadingDataObject = p_LoadingDataObject;
    }
```

```
private:
```

```
    float& (SCDLLCALL* Internal_GetChartBarValue)(void* p_LoadingDataObject, int SubgraphIndex, int BarIndex);
```

```
    const SCDatetime& (SCDLLCALL* Internal_GetChartBarDateTime)(void* p_LoadingDataObject, int BarIndex);
```

```
    SCInputRef & (SCDLLCALL* Internal_GetInput)(void* p_LoadingDataObject, int InputIndex);
```

```
    int& (SCDLLCALL* Internal_GetPersistentInt)(void* p_LoadingDataObject, int Key);
```

```
    float& (SCDLLCALL* Internal_GetPersistentFloat)(void* p_LoadingDataObject, int Key);
```

```
    double& (SCDLLCALL* Internal_GetPersistentDouble)(void* p_LoadingDataObject, int Key);
```

```
    int64_t& (SCDLLCALL* Internal_GetPersistentInt64)(void* p_LoadingDataObject, int Key);
```

```
    SCDatetime& (SCDLLCALL* Internal_GetPersistentSCDateTime)(void* p_LoadingDataObject, int Key);
```

```
    void* m_p_LoadingDataObject;
```

```
public:
```

```
    //Additional Input members
```

```
    float BidPrice; //The bid price associated with the last trade in NewFileRecord
```

```
    float AskPrice; //The ask price associated with the last trade in NewFileRecord
```

```

const c_VAPContainer *VolumeAtPriceForBars;

private:
int (SCDLLCALL* InternalFormattedEvaluate)(float Value1, unsigned int Value1Format,
OperatorEnum Operator,
float Value2, unsigned int Value2Format,
float PrevValue1, float PrevValue2,
int* CrossDirection) = nullptr;
};
/*=====*/
inline float& s_CustomChartBarInterface::GetChartBarValue(int SubgraphIndex, int BarIndex)
{
return Internal_GetChartBarValue(m_p_LoadingDataObject, SubgraphIndex, BarIndex);
}
/*=====*/
inline const SCDateTime& s_CustomChartBarInterface::GetChartBarDateTime(int BarIndex)
{
return Internal_GetChartBarDateTime(m_p_LoadingDataObject, BarIndex);
}
/*=====*/
inline SCInputRef& s_CustomChartBarInterface::GetInput(int InputIndex)
{
return Internal_GetInput(m_p_LoadingDataObject, InputIndex);
}
/*=====*/

inline int& s_CustomChartBarInterface::GetPersistentInt( int Key)
{
return Internal_GetPersistentInt(m_p_LoadingDataObject, Key);
}
/*=====*/
inline float& s_CustomChartBarInterface::GetPersistentFloat(int Key)
{
return Internal_GetPersistentFloat(m_p_LoadingDataObject, Key);
}

}
/*=====*/
inline double& s_CustomChartBarInterface::GetPersistentDouble(int Key)
{
return Internal_GetPersistentDouble(m_p_LoadingDataObject, Key);
}

}
/*=====*/
inline int64_t& s_CustomChartBarInterface::GetPersistentInt64(int Key)
{
return Internal_GetPersistentInt64(m_p_LoadingDataObject, Key);
}
}
/*=====*/
inline SCDateTime& s_CustomChartBarInterface::GetPersistentSCDateTime(int Key)
{
return Internal_GetPersistentSCDateTime(m_p_LoadingDataObject, Key);
}
}
/*=====*/

typedef s_CustomChartBarInterface& SCCustomChartBarInterfaceRef;

typedef void (SCDLLCALL* fp_ACSCustomChartBarFunction)(SCCustomChartBarInterfaceRef);

#endif

```