Component Interface for Version 4 of PC-SPAN, SPAN Risk Manager, and SPAN Risk Manager Clearing

A **real-time component interface** is available for PC-SPAN version 4. With this additional software module, licensed separately as SPAN Real-Time Component Interface (SPAN RTCI), you can develop programs which interface to PC-SPAN in real-time.

For example, suppose a customer requests an order. Using the component interface, your program can call PC-SPAN, specify the updated position that will result from the order, request that the performance bond be calculated, and receive back the updated performance bond requirement -- all with extremely rapid execution times.

The SPAN Risk Manager Clearing software also has a real-time component interface. With this software module, in addition to everything you can do with the component interface to PC-SPAN, you can develop programs which execute the functions of the SPAN Risk Manager Clearing in real-time.

For example, you could programmatically update market prices for futures and physicals, and then recalculate volatilities, theoretical prices and risk arrays, with extremely fast.

This topic provides a technical reference to the real-time component interfaces for PC-SPAN, SPAN Risk Manager, and SPAN Risk Manager Clearing.

- dispinterface ISpanCom
  - short Load(BSTR fileName, boolean replaceExc, boolean useExtLayout);
  - short Save(BSTR fileName);
  - short SelectPIT(BSTR busDate, short isSettle, short isFinal, BSTR busTime, BSTR description);
  - short SelectPortfolio(BSTR firmCode, BSTR acctCode, BSTR segType);
  - void ResetPIT();
  - void ResetPortfolio();
  - short Calculate();
  - double GetPortfSpanReq(short pbClass, boolean isInitial);
  - double GetPortfTotalReq(short pbClass, boolean isInitial);
  - double GetCurPortfSpanReq(short pbClass, boolean isInitial, BSTR curCode);
  - double GetCurPortfTotalReq(short pbClass, boolean isInitial, BSTR curCode);
  - void Delete();
  - double GetPortfLongFutValue();
  - double GetPortfShortFutValue();
  - double GetPortfLongOptValue();
  - double GetPortfShortOptValue();
  - void SetLogLevel(short processId, short priorityId);
  - void LogSave(BSTR fileName);
  - void LogClear();
  - short CreatePortfolio(BSTR firmCode, BSTR acctCode, BSTR segType);
  - short SetPortfAcctType(BSTR acctType, boolean isClearing);
  - short SetPortfParent(BSTR firmCode, BSTR acctCode, BSTR segType);
  - short SetPortfCurrency(BSTR currencyCode);
  - BSTR GetPortfCurrency();
  - short SetPortfLedgerBalance(double value);
  - double GetPortfLedgerBalance();
  - short SavePortfolios(BSTR filename);
  - short SetPortfOpenTradeEquity(double value);
  - double GetPortfOpenTradeEquity();
  - short SetPortfSecurityOnDeposit(double value);
  - double GetPortfSecurityOnDeposit();
  - short SetPortfPosition(BSTR exchCmplxAcro, BSTR exchAcro, BSTR pfCode, short pfType, BSTR futPeriod, BSTR optPeriod, boolean isPut, double strike, long totalLong, long totalShort, long intraLong, long intraShort, long interLong, long interShort, long nakedLong, longnakedShort);
  - short ChangePortfPosition(BSTR exchCmplxAcro, BSTR exchAcro, BSTR pfCode, short pfType, BSTR futPeriod, BSTR optPeriod, boolean isPut, double strike, long totalLong, long totalShort, long intraLong, long intraShort, long interLong, long interShort, long nakedLong, longnakedShort);
  - short SavePositions(BSTR fileName);
  - short LoadStream(SAFEARRAY(BYTE) data);
- dispinterface ISpanComRM
  - short SelectExchangeComplex(BSTR excAcro);
  - short SelectBFCC(BSTR ccCode);
  - short SelectProductFamily(BSTR exchCode, BSTR pfCode, short pfType);
  - void ResetExchangeComplex();
  - void ResetBFCC();
  - void ResetProductFamily();
  - short CalcImpliedVolatility(boolean flatCabVol);
  - short CalcCallPutAverage(boolean outMoney, boolean acceptZero);
  - short CalcSeriesVolatility(boolean useZero, double minExpTime, short maxIn, short maxOut, short minAccept);
  - short CalcPrice(short meth, boolean resetAll);
  - short CalcRiskArray(short meth, short limitMeth, boolean useTheorPrice);
  - void UpdatePrice(double value, short meth);
  - void UpdateVolatility(double value, short meth);
  - void UpdatePriceScan(double value, short meth);
  - void UpdateVolScan(double value, short meth);
  - void UpdateRiskFreeRate(double value, short meth);
  - void UpdateDividendYield(double value, short meth);

- void UpdateTimeToExpiration(double value, short meth);
- short UpdateCDSValue(short valueType, double value, short meth);
- short SetStartPeriod(BSTR periodCode);
- short SetEndPeriod(BSTR periodCode);
- void ResetPeriods();
- short CopyPIT(BSTR description, BSTR busDate, short isSettle, short isFinal, BSTR busTime);
- void DoMarketObservation();
- void ResetMarketPrices();
- double GetPortfMTMFut();
- double GetPortfMTMPrem();
- double GetPortfMTMTFut();
- double GetPortfMTMTPrem();
- double GetPortfUnrealizedPL();
- double GetPortfRealizedPLOpen();
- double GetPortfRealizedPLLiq();
- short SaveRegistryTo(BSTR fileName);
- short ApplyWhatIf(BSTR fileName, boolean doCalc);
- short CalcValues();
- short CalcGreeks(short meth);
- short ReplicateBasePrices();
- short ApplyVolatilitySkew(short meth);
- short PostTrade(BSTR timeStamp, long tradeId, BSTR exchCmplxAcro, BSTR exchAcro, BSTR pfCode, short pfType, BSTR futPeriod, BSTR optPeriod, boolean isPut, double strike, double price, long tradeQty);
- short ResetPrices(short prodType);

Below is an example of calling the component interface from Visual C++ code.

```
#include <stdio.h>
#include <ole2.h>
#import "C:\Span4\Bin\SpanCom.tlb" no_namespace, named_guids

int main(int argc, char* argv[])
{

	if(FAILED(::CoInitialize(NULL))) return 1;

	ISpanCom* pSpanCom;
	if(SUCCEEDED(::CoCreateInstance(CLSID_SpanCom, NULL, CLSCTX_LOCAL_SERVER,
	__uuidof(ISpanCom), (LPVOID*) &pSpanCom)))
	{
		printf( "Success\n");
		// pSpanCom->Load("C:\\cme0216s.par",1,0);
		pSpanCom->Release();
	}

	else printf("Fail\n");
	::CoUninitialize();
	return 0;
}
```

---

dispinterface ISpanCom

methods:

short Load(BSTR *fileName*, boolean *replaceExc*, boolean *useExtLayout*);

Description:

Load any SPAN recognized file into PC-SPAN, including SPAN files (*.SPN), classic "flat-files" (*.PA2), position files (*.txt and *.pos), etc. All position files (*.txt and *.pos) require corresponding risk parameter files to be already loaded. Positions are loaded into the currently selected point in time. If this point in time is not present positions are loaded into the first available point in time.

For more details on supported file formats, click here.

Parameters:

- *fileName* - BSTR - The document to be loaded.
- *replaceExc* - bool - Specifies whether PC-SPAN should replace already loaded exchange complexes with the ones from the file (true) or leave them intact (false)
- *useExtLayout* - bool - Specifies whether PC-SPAN should use extended file layout when loading positions. It is only used for loading old format position files (*.txt).

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short Save(BSTR *fileName*);

Description:

Saves all the information from PC-SPAN into a file.

Parameters:

- *fileName* - BSTR - The name of document to be saved, in the SPAN XML format.

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short SelectPIT(BSTR *busDate*, short *isSettle*, short *isFinal*, BSTR *busTime,* BSTR *description*);

Description:

Selects a specified point in time as current point in time. Once selected, a point in time remains selected until it is reset, a different point in time is selected, or until the point in time is cleared.

Parameters:

- *busDate* - BSTR - The date of the point in time to select. The point in time must be in a "YYYYMMDD" format. Alternatively, the busDate string parameter may be empty to select the first point in time.
- *isSettle* - short - Search for a point in time where the settlement is: '0' for non-settlement; '1' for settlement; or '-1' for no preference on settlement indication.
- *isFinal* - short - Search for a point in time where the final flag is: '0' for not final; '1' for final; or '-1' for no preference on final indication.
- *busTime* - BSTR - When the *isSettle* flag is 0, the busTime may be optionally specified. It is formatted as "HHMM". If *isSettle* flag is not 0, the busTime is ignored.
- *description* - BSTR - When the *isSettle* flag is not -1, the description may be optionally specified to match the description of available points in time. If passed as an empty string, it is ignored.

Remarks:

If *busDate* is empty and the *isSettle* is -1, the rest of the parameters are ignored and first available point in time is selected. If either *isSettle* or *isFinal* is set to –1 this parameter and all parameters following it are ignored. If *busTime* or *description* is empty it is ignored.

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short SelectPortfolio(BSTR *firmCode*, BSTR *acctCode*, BSTR *segType*);

Description:

Selects specified portfolio as current. This method searches for the specified portfolio in the currently selected point in time. If point in time is not selected it fails.

Parameters:

- *firmCode* - BSTR - If not empty, it will be used to match the firm code to find the portfolio.
- *acctCode* - BSTR - If not empty, it will be used to match the account code.
- *segType* - BSTR - If not empty, it will be used to match the seg type code.

Remarks:

If all parameters are empty, the first portfolio will be selected. Any combination of parameters may be used.

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

void ResetPIT();

Description:

Sets currently selected point in time to NULL (removes selection). Also resets portfolio selection.

Parameters:

None

Remarks:

When the point in time selection is reset, the point in time is still available. It is merely no longer selected as the active point in time.

Return Value:

None

void ResetPortfolio();

Description:

Sets currently selected portfolio to NULL (removes selection).

Parameters:

None

Remarks:

When this call is made, the portfolio is still available. It is no longer selected as the active portfolio. The point in time selection is not affected by this call.

Return Value:

None

short Calculate();

Description:

Performs SPAN requirements calculations for the currently selected object. If portfolio selection is not NULL it does calculations only for the selected portfolio. If it is NULL and point in time selection is not NULL it does calculations for all portfolios in the selected point in time. If both selected portfolio and selected point in time are NULL it does calculations for all points in time and portfolios loaded into PC-SPAN.

Parameters:

None

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

double GetPortfSpanReq(short *pbClass*, boolean *isInitial*);

Description:

Returns SPAN requirement calculated for the selected portfolio. If a portfolio is not selected (NULL) or if Calculate has not been called, the return is 0.

Parameters:

- *pbClass* - short - Specifies the class of the requirement. It can be 1(CORE) or 2(RESERVE).
- *isInitial* - boolean - Indicates maintenance (false) or initial (true) requirement.

Return Value:

double - Returns SPAN requirement calculated for the selected portfolio.

double GetPortfTotalReq(short *pbClass*, boolean *isInitial*);

Description:

Returns Total requirement calculated for the selected portfolio. If a portfolio is not selected (NULL) or if Calculate has not been called, the return is 0.

Parameters:

- *pbClass* - short - Specifies the class of the requirement. It can be 1(CORE) or 2(RESERVE).
- *isInitial* - boolean - Indicates maintenance (false) or initial (true) requirement.

Return Value:

double - Returns Total requirement calculated for the selected portfolio.

double GetCurPortfSpanReq(short *pbClass*, boolean *isInitial*, BSTR *curCode*);

Description:

Returns SPAN requirement calculated for the selected portfolio for the specified currency. If a portfolio is not selected (NULL), or if an invalid curCode is passed, or if Calculate has not been called, the return is 0.

Parameters:

- *pbClass* - short - Specifies the class of the requirement. It can be 1(CORE) or 2(RESERVE).
- *isInitial* - boolean - Indicates maintenance (false) or initial (true) requirement.
- *curCode* - BSTR - Specifies the currency code to get the Span requirement in.

Return Value:

double - Returns SPAN requirement calculated for the selected portfolio for the specified currency.

double GetCurPortfTotalReq(short *pbClass*, boolean *isInitial*, BSTR *curCode*);

Description:

Returns Total requirement calculated for the selected portfolio for the specified currency. If a portfolio is not selected (NULL), or if an invalid curCode is passed, or if Calculate has not been called, the return is 0.

Parameters:

- *pbClass* - short - Specifies the class of the requirement. It can be 1(CORE) or 2(RESERVE).
- *isInitial* - boolean - Indicates maintenance (false) or initial (true) requirement.
- *curCode* - BSTR - Specifies the currency code to get the Total requirement in.
- *curCode* - BSTR - Specifies the currency code to get the Total requirement in.

Return Value:

double - Returns Total requirement calculated for the selected portfolio for the specified currency.

void Delete();

Description:

Deletes currently selected object from PC-SPAN. If a portfolio selection has been made, it deletes only the selected portfolio. If it is not selected (NULL) and a point in time selection is selected (NULL) it deletes the selected point in time. If both neither a portfolio or a point in time are selected (both NULL), then the whole contents of the PC-SPAN is deleted.

Parameters:

None

Return Value:

None

double GetPortfLongFutValue();

Description:

Returns long futures value calculated for the selected portfolio. If portfolio is not selected (NULL) it returns 0.

Parameters:

None

Return Value:

double - Returns long futures value for selected portfolio.

double GetPortfShortFutValue();

Description:

Returns short futures value calculated for the selected portfolio. If portfolio is not selected (NULL) it returns 0.

Parameters:

None

Return Value:

double - Returns short futures value for selected portfolio.

double GetPortfLongOptValue();

Description:

Returns long options value calculated for the selected portfolio. If portfolio is not selected (NULL) it returns 0.

Parameters:

None

Return Value:

double - Returns long options value for selected portfolio.

double GetPortfShortOptValue();

Description:

Returns short options value calculated for the selected portfolio. If portfolio is not selected (NULL) it returns 0.

Parameters:

None

Return Value:

double - Returns short options value for selected portfolio.

void SetLogLevel(short *processId*, short *priorityId*);

Description:

Use this method to specify logging level for PC-SPAN.

Parameters

- *processId* - short - The upper limit for process id of the log message source. Possible values for processLimits are:

PROC_ID_LOAD_RISK	= 10;
PROC_ID_LOAD_POS	= 20;
PROC_ID_LOAD_XML	= 30;
PROC_ID_OMNIBUS	= 40;
PROC_ID_CALC_START	= 51;
PROC_ID_CALC_PBOND	= 60;
PROC_ID_CALC_SPREAD	= 70;
PROC_ID_CALC_LIQ_RISK	= 80;
PROC_ID_CALC_DETAIL_START	= 101;
PROC_ID_CALC_SPREAD_DETAIL	= 110;
PROC_ID_CALC_LIQ_RISK_DETAIL	= 120;
PROC_ID_CALC_POS_DETAIL	= 130;
PROC_ID_UPPER	= -1;

- *priorityId* - short - The lower limit for the log message priority. Possible values for the priorityId are:

PRIORITY_ID_LOW	= -1;
PRIORITY_ID_NORMAL	= 10;
PRIORITY_ID_WARNING	= 20;
PRIORITY_ID_ERROR	= 30;

Remarks:

Default values are 0 (processId) and 30 (priorityId - only errors) are logged.

Return Value:

None

void LogSave(BSTR *fileName*);

Description:

Saves PC-SPAN log into the file specified by *fileName*.

Parameters:

fileName - BSTR - The file name to save the log as.

Return Value:

None

void LogClear();

Description:

Clears the PC-SPAN Log.

Parameters:

None

Return Value:

None

short CreatePortfolio(BSTR *firmCode*, BSTR *acctCode*, BSTR *segType*);

Description:

Creates a new portfolio with specified parameters in the selected point in time and selects it as current. The call will fail if a point in time is not selected. If portfolio with given parameters already exist selects the existing portfolio as current (the call does not create a duplicate one).

Parameters:

- *firmCode* - BSTR - The firm code to create a portfolio with.
- *acctCode* - BSTR - The account code to create a portfolio with.
- *segType* - BSTR - The segment type to create a portfolio with. If *segType* is empty, a “N/A” segment type is assumed.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short SetPortfAcctType(BSTR *acctType*, boolean *isClearing*);

Description:

Sets account type for the selected portfolio. If portfolio is not selected the call will fail. If the specified account type is not found in the organization master database the call will fail. The selected portfolio can not have positions already defined otherwise the call will fail.

Parameters:

- *acctType* - BSTR - The account type for the selected portfolio. The following values are currently defined for the *acctType*:

N	- normal
M	- member
H	- hedge
S	- speculator
O	- omnibus (speculator)
Q	- omnibus (hedge).

- *isClearing* - boolean - specifies whether account is customer(false) or clearing(true).

Return Value:

short - Returns 0 (success) or non-zero (failure).

short SetPortfParent(BSTR *firmCode*, BSTR *acctCode*, BSTR *segType*);

Description:

Sets parent portfolio for the selected portfolio by finding a portfolio using the parameters provided. Used only for omnibus accounts. Only omnibus account can be a parent of another portfolio. Only one level of parents is supported. Fails if portfolio is not selected, parent portfolio can not be found or omnibus rules are violated.

Parameters:

- *firmCode* - BSTR - The portfolio firmCode to search for the selected portfolio.
- *acctCode* - BSTR - The portfolio acctCode to search for the selected portfolio.
- *segType* - BSTR - The portfolio segType to search for the selected portfolio.

Return Value

short - Returns 0 (success) or non-zero (failure).

short SetPortfCurrency(BSTR *currencyCode*);

Description:

Sets currency for the selected portfolio.

Parameters

- *currencyCode* - BSTR - A three-letter ISO code for the currency (USD, EUR, GBP etc.).

Remarks

The call will fail if a portfolio is not selected or if the currency parameter is not defined in the organization master database.

Return Value

short - Returns 0 (success) or non-zero (failure).

BSTR GetPortfCurrency();

Description:

Returns a three-letter ISO code for the currency of the selected portfolio. If a portfolio is not selected or the currency for it is not specified method returns an empty string.

Parameters:

None

Return Value:

BSTR - Returns an ISO code for the selected portfolio currency.

short SetPortfLedgerBalance(double *value*);

Description:

Sets ledger balance for the selected portfolio. Fails if portfolio is not selected.

Parameters:

- *value* - double - The portfolio ledger balance to set.

Return Value:

short - Returns 0 (success) or non-zero (failure).

double GetPortfLedgerBalance();

Description:

Returns ledger balance for the selected portfolio. If portfolio is not selected, the call will fail.

Parameters:

None

Return Value:

double - Returns the ledge balance or 0 if no portfolio is selected.

short SavePortfolios(BSTR *filename*);

Description:

Saves portfolios to a file.

Parameters:

- *fileName* - BSTR - The name of the portfolio file to save.

Remarks:

If a portfolio is selected, only the selected portfolio will be saved. If a portfolio is not selected but a point in time is selected, all of the portfolios belonging to the selected point in time will be saved. A portfolio or a point in time must be selected, otherwise the call will fail.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short SetPortfOpenTradeEquity(double *value*);

Description:

Sets open trade equity for the selected portfolio. Fails if portfolio is not selected.

Parameters:

- *value*- double - The open trade equity to set.

Return Value:

short - Returns 0 (success) or non-zero (failure).

double GetPortfOpenTradeEquity();

Description:

Returns open trade equity for the selected portfolio. If a portfolio is not selected, the return will be 0.

Parameters:

None

Return Value:

double - Returns open trade equity or 0 if no portfolio is selected.

short SetPortfSecurityOnDeposit(double *value*);

Description:

Sets security on deposit for the selected portfolio. The call will fail if a portfolio is not selected.

Parameters:

- *value* - double - The security on deposit value to set.

Return Value:

short - Returns 0 (success) or non-zero (failure).

double GetPortfSecurityOnDeposit();

Description:

Returns security on deposit for the selected portfolio. If a portfolio is not selected returns 0.

Parameters:

None

Return Value:

double - Returns the security on deposit or 0 if no portfolio is selected.

short SetPortfPosition(BSTR *exchCmplxAcro*, BSTR *exchAcro*, BSTR *pfCode*, short *pfType*, BSTR *futPeriod*, BSTR *optPeriod*, boolean *isPut*, double *strike*, long *totalLong*, long *totalShort*, long *intraLong*, long *intraShort*, long *interLong*, long *interShort*, long *nakedLong*, long*nakedShort*);

Description:

Sets positions in the selected portfolio in the specified contract. If portfolio is not selected or contract can not be found the method will fail.

Parameters:

- *exchCmplxAcro* - BSTR - The exchange complex acronym (“CME”, “BOTCC”, etc.)
- *exchAcro* - BSTR - The exchange acronym (“CME”, “CBT”, etc.)
- *pfCode* - BSTR - The product family code (“SP”, “BP”, etc.)
- *pfType* - short - The product family type. Possible values are:

1	- physical
2	- debt
3	- stock
11	- future
12	- equivalence debt
14	- forward
15	- interest rate swap
16	- E-Debt
17	- credit rate swap
21	- option on physical
22	- option on future
23	- option on equity
31	- combination
41	- option on combination

- *futPeriod* - BSTR - The period code for future contract (period code for the underlying contract for options). Format is usually YYYYMM, there are special cases like weeklies and flexes when some characters are added to this format (for example “200002W3”). For physicals, equity and options on physicals/equity this parameter should be set to “000000”.
- *optPeriod* - BSTR - The period code for option contract. This parameter and following contract identification parameters are ignored for non-option contracts. Same format as *futPeriod*.
- *isPut* - boolean - The put (true) or call (false) flag for options.
- *strike* - double - The strike price for options. For CDS products, this is the coupon.
- *totalLong* - long - The new value for total long positions. If portfolio is net based this value will be used for calculating net positions (**Net** = *totalLong* – *totalShort*)
- *totalShort* - long - The new value for total short positions. If portfolio is net based this value will be used for calculating net positions (**Net** = *totalLong* – *totalShort*)
- *intraLong* - long - The new value for intra long positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedLong** = *totalLong* – *intraLong* - *interLong*)
- *intraShort* - long - The new value for intra short positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedShort** = *totalShort* – *intraShort* - *interShort*)
- *interLong* - long - The new value for inter long positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedLong** = *totalLong* – *intraLong* - *interLong*)
- *interShort* - long - The new value for inter short positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedShort** = *totalShort* – *intraShort* - *interShort*)
- *nakedLong* - long - The new value for naked long positions. If portfolio is net based this value is ignored.
- *nakedShort* - long - The new value in naked short positions. If portfolio is net based this value is ignored.

Remarks:

If number of positions specified is inconsistent *totalLong* < *intraLong* + *intraShort* or *totalShort* < *intraShort* + *interShort*, then method fails. If a portfolio hasn't been selected, the method will fail.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short ChangePortfPosition(BSTR *exchCmplxAcro*, BSTR *exchAcro*, BSTR *pfCode*, short *pfType*, BSTR *futPeriod*, BSTR *optPeriod*, boolean *isPut*, double *strike*, long *totalLong*, long *totalShort*, long *intraLong*, long *intraShort*, long *interLong*, long *interShort*, long *nakedLong*, long*nakedShort*);

Description:

Modifies positions in the selected portfolio in the specified contract. If portfolio is not selected or contract can not be found method fails.

Parameters:

- *exchCmplxAcro* - BSTR - The exchange complex acronym (“CME”, “BOTCC”, etc.)
- *exchAcro* - BSTR - The exchange acronym (“CME”, “CBT”, etc.)
- *pfCode* - BSTR - The product family code (“SP”, “BP”, etc.)
- *pfType* - short - The product family type. Possible values are:

1	- physical
2	- debt
3	- stock
11	- future
12	- equivalence debt
14	- forward
15	- interest rate swap
16	- E-Debt
17	- credit rate swap
21	- option on physical
22	- option on future
23	- option on equity
31	- combination
41	- option on combination

- *futPeriod* - BSTR - The period code for future contract (period code for the underlying contract for options). Format is usually YYYYMM, there are special cases like weeklies and flexes when some characters are added to this format (for example “200002W3”). For physicals, equity and options on physicals/equity this parameter should be set to “000000”.
- *optPeriod* - BSTR - The period code for option contract. This parameter and following contract identification parameters are ignored for non-option contracts. Same format as *futPeriod*.
- *isPut* - boolean - The put (true) or call (false) flag for options.

- *strike* - double - The strike price for options. For CDS products, this is the coupon.
- *totalLong* - long - The change in total long positions. If portfolio is net based this value will be used for calculating change in net positions (**Net** = *totalLong – totalShort*)
- *totalShort* - long - The change in total short positions. If portfolio is net based this value will be used for calculating net positions (**Net** = *totalLong – totalShort*)
- *intraLong* - long - The change in intra long positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedLong** = *totalLong – intraLong - interLong*)
- *intraShort* - long - The change in intra short positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedShort** = *totalShort – intraShort - interShort*)
- *interLong* - long - The change in inter long positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedLong** = *totalLong – intraLong - interLong*)
- *interShort* - long - The change in inter short positions. If portfolio is net based this value is ignored. It is used for calculating number of naked positions (**NakedShort** = *totalShort – intraShort - interShort*)
- *nakedLong* - long - The change in naked long. If portfolio is net based this value is ignored.
- *nakedShort* - long - The change in naked short. If portfolio is net based this value is ignored.

Remarks:

If number of positions specified is inconsistent *totalLong* < *intraLong* + *intraShort* or *totalShort* < *intraShort* + *interShort*, then method fails. If a portfolio hasn't been selected, the method will fail.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short SavePositions(BSTR *fileName*);

Description:

Saves positions for the selected object into the file, specified by *fileName*.

Parameters:

- *fileName* - BSTR - The file name of the position file to save.

Remarks:

If selected portfolio is not NULL saves only positions for selected portfolio. If selected portfolio is NULL, but selected point in time is not saves positions for all portfolios in the selected point in time. If both selected portfolio and selected point in time are NULL method fails.

Return Value:

short - Returns 0 (success) or non-zero (failure).
short SaveCDSPositions(BSTR *fileName*);

Description:

Credit editions of SPAN only. Saves CDS positions for the selected object in to the file specified by *fileName*.

Parameters:

- *fileName* - BSTR - The file name of the CDS position file to save.

Remarks:

This call is applicable to PC-Credit only. If selected portfolio is not NULL saves only positions for selected portfolio. If selected portfolio is NULL, but selected point in time is not saves positions for all portfolios in the selected point in time. If both selected portfolio and selected point in time are NULL method fails.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short LoadStream(SAFEARRAY(BYTE) *data*);

Description:

Similar to the Load method, this method will load document passed directly in through the *data* parameter byte array. This method can bypass the need to save a newly constructed document to disk for the sole purpose of loading it in to SPAN.

Parameters:

- *data* - SAFEARRAY(BYTE) - The stream of data representing a SPAN data document, such as a position file, to load in to SPAN.

Return Value:

short - Returns 0 (success) or non-zero (failure).

dispinterface ISpanComRM

methods:

short SelectExchangeComplex(BSTR *excAcro*);

Description:

Selects specified exchange complex as current. *excAcro* is an acronym used to identify exchange complex. A Point In Time should be already selected prior to calling this method. Call to this method automatically resets BFCC and product family selection.

Parameters:

excAcro - BSTR - The exchange complex to select.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short SelectBFCC(BSTR *ccCode*);

Description:

Selects specified BFCC as current. Exchange Complex should be already selected prior to calling this method. Call to this method automatically resets product family selection.

Parameters:

- *ccCode* - BSTR - The combined commodity code to select.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short SelectProductFamily(BSTR *exchCode*, BSTR *pfCode*, short *pfType*);

Description:

Selects specified product family as current.

Parameters:

- *exchAcro* - BSTR - The exchange acronym (“CME”, “CBT”, etc.)
- *pfCode* - BSTR - The product family code (“SP”, “BP”, etc.)
- *pfType* - short - The product family type. Possible values are:

1	- physical
2	- debt
3	- stock
11	- future
12	- equivalence debt
14	- forward
15	- interest rate swap
16	- E-Debt
17	- credit rate swap
21	- option on physical
22	- option on future
23	- option on equity
31	- combination
41	- option on combination

Remarks:

BFCC should be already selected prior to calling this method.

Return Value:

short - Returns 0 (success) or non-zero (failure).

void ResetExchangeComplex();

Description:

Sets currently selected exchange complex to NULL (removes selection). Also resets BFCC and Product Family selection.

Parameters:

None

Return Value:

None

void ResetBFCC();

Description:

Sets currently selected BFCC to NULL (removes selection). Also resets Product Family selection.

Parameters:

None

Return Value:

None

void ResetProductFamily();

Description:

Sets currently selected product family to NULL (removes selection).

Parameters:

None

Return Value:

None

short CalcImpliedVolatility(boolean *flatCabVol*);

Description:

Calculates Implied Volatilities for selected object. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time.

Parameters:

- *flatCabVol* - boolean - Indicator to calculate volatility for cabs for option series that have a cab price type.

Return Value:

short - Returns 0 (success) or non-zero (failure).

short CalcCallPutAverage(boolean *outMoney*, boolean *acceptZero*);

Description:

Does Call/Put Averaging for selected object. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time.

Parameters:

- *outMoney* - boolean - true means use out-of-money volatility, false means do averaging
- *acceptZero* - boolean - true means accept zero volatility as valid, false is the opposite

Return Value:

short - Returns 0 (success) or non-zero (failure).

short CalcSeriesVolatility(boolean *useZero*, double *minExpTime*, short *maxIn*, short *maxOut*, short *minAccept*);

Description:

Calculates Series Level Volatilities for selected object. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time.

Parameters:

- *useZero* - boolean - An indication of true means do not filter out zero implied volatilities during calculations, false - filter them out.
- *minExpTime* - double - The minimum time to expiration in years. Standard value used by most algorithms is 0.0273973 (10 business days).
- *maxIn* - short - The number of closest in the money options which volatilities will be used in calculations (1 is standard).
- *maxOut* - short - The number of closest out of the money options which volatilities will be used in calculations (3 is standard).
- *minAccept* - short - minimum sufficient number of around the money volatilities (5 is standard).

Return Value:

short - Returns 0 (success) or non-zero (failure).

short CalcPrice(short *meth*, boolean *resetAll*);

Description:

Calculates Theoretical Prices for selected object. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time.

Parameters:

- *meth* - boolean - The calculation method code:

1	- Use option/series level volatility
2	- Use option level volatility only
3	- Use series level volatility only

- *resetAll* - An indication of true will reset prices for all contracts, false will reset prices only for contracts with undefined price

Return Value:

short - Returns 0 (success) or non-zero (failure).

short CalcRiskArray(short *meth*, short *limitMeth*, boolean *useTheorPrice*);

Description:

Calculates Risk Arrays for selected object. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time.

Parameters:

- *meth* - short - The calculation method code:

1	- Use option/series level volatility
2	- Use option level volatility only
3	- Use series level volatility only

- *limitMeth* - short - The method of how risk array values will be limited:

0	- Regular limit
1	- Limit Loss Option
2	- Limit Gain Option
3	- Limit Option
4	- Limit Range
5	- Limit Loss Option Range
6	- Limit Gain Option Range
7	- Limit Option Range

- *useTheorPrice* - boolean - An indication of true means to use theoretical prices.

Return Value:

short - Returns 0 (success) or non-zero (failure).

void UpdatePrice(double *value*, short *meth*);

Description:

Use this method to Update Prices. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update price using *value* and one of the methods below.

Parameters:

- *value* - double - The price value
- *meth* - short - The method to use for updating the price. Possible values for *meth*:

0	(SET) - Set price to *value*.
1	(CHG) - Change price by *value*
2	(PCNT) - Change price by *value* %

Return Value:

None

void UpdateVolatility(double *value*, short *meth*);

Description:

Use this method to Update Volatilities. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update volatilities using *value* and one of the methods below.

Parameters:

- *value* - double - The volatility value
- *meth* - short - The method to use for updating the volatility. Possible values for *meth*:

0	(SET) - Set volatility to *value*.
1	(CHG) - Change volatility by *value*
2	(PCNT) - Change volatility by *value* %

Return Value:

None

void UpdatePriceScan(double *value*, short *meth*);

Description:

Use this method to Update Price Scans. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update price scans using *value* and one of the methods below.

Parameters:

- *value* - double - The price scan value
- *meth* - short - The method to use for updating the price scans. Possible values for *meth*:

0	(SET) - Set price scan to *value*.
1	(CHG) - Change price scan by *value*
2	(PCNT) - Change price scan by *value* %

Return Value:

None

void UpdateVolScan(double *value*, short *meth*);

Description:

Use this method to Update Volatility Scans. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update volatility scans using *value* and one of the methods below.

Parameters:

- *value* - double - The volatility scan value
- *meth* - short - The method to use for updating the volatility scans. Possible values for *meth*:

0	(SET) - Set volatility scan to *value*.
1	(CHG) - Change volatility scan by *value*
2	(PCNT) - Change volatility scan by *value* %

Return Value:

None

void UpdateRiskFreeRate(double *value*, short *meth*);

Description:

Use this method to Update Risk-Free Rates. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update risk free rates using *value* and one of the methods below.

Parameters:

- *value* - double - The risk free rate value
- *meth* - short - The method to use for updating the risk free rates. Possible values for *meth*:

0	(SET) - Set risk free rate to *value*.
1	(CHG) - Change risk free rate by *value*
2	(PCNT) - Change risk free rate by *value* %

Return Value:

None

void UpdateDividendYield(double *value*, short *meth*);

Description:

Use this method to Update Dividend Yields. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update dividend yields using *value* and one of the methods below.

Parameters:

- *value* - double - The dividend yield value
- *meth* - short - The method to use for updating the dividend yields. Possible values for *meth*:

0	(SET) - Set dividend yield to *value*.
1	(CHG) - Change dividend yield by *value*
2	(PCNT) - Change dividend yield by *value* %

Return Value:

None

void UpdateTimeToExpiration(double *value*, short *meth*);

Description:

Use this method to Update Times to Expiration. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update time to expiration using *value* and one of the methods below.

Parameters:

- *value* - double - The time to expiration value
- *meth* - short - The method to use for updating the times to expiration. Possible values for *meth*:

0	(SET) - Set time to expiration to *value*.
1	(CHG) - Change time to expiration by *value*
2	(PCNT) - Change time to expiration by *value* %

Return Value:

None

short UpdateCDSValue(short *valueType*, double *value*, short *meth*);

Description:

Use this method to Update credit default swap values. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time. It will update a credit default swap value using *value* and one of the methods below.

Parameters:

- *valueType* - short - The specific credit default swap shock to update. Possible values for *valueType*:

1	- Systematic shock - Applies to combined commodity level
2	- Convergence shock (IG_COMP) - Applies to Exchange Complex level
3	- Divergence shock (HY_COMP) - Applies to Exchange Complex level
5	- Sector shock - Applies to combined commodity level

- *value* - double - The credit default swap value
- *meth* - short - The method to use for updating the credit default swap values. Possible values for *meth*:

0	(SET) - Set dividend yield to *value*.
1	(CHG) - Change dividend yield by *value*
2	(PCNT) - Change dividend yield by *value* %

Return Value:

short - Returns 0 (a value was successfully updated) or non-zero (failure).

short SetStartPeriod(BSTR *periodCode*);

Description:

Filters the start period of the selected product family

Parameters:

- *periodCode* - BSTR - The period code can be specified when a product family is selected to filter for period codes of contracts and series. When an empty string is passed, the start period code is cleared. The filter can be used in the following calculate and update methods:
  - CalcImplitedVolitility
  - CalcCallPutAverage
  - CalcPrice
  - CalcGreeks
  - CalcRiskArray
  - UpdatePrice
  - UpdateVolatility
  - UpdatePriceScan
  - UpdateVolScan
  - UpdateRiskFreeRate
  - UpdateDividendYield
  - UpdateTimeToExpiration

Remarks:

A product family must be selected. The start period code will be cleared when a blank string is passed, or if one of the following methods is called: ResetPeriods, SelectPIT, ResetPIT, Delete, SelectExchangeComplex, SelectBFCC, SelectProductFamily, ResetExchangeComplex, ResetBFCC, or ResetProductFamily

Return Value:

short - Returns 0 (success) or non-zero (failure - PF not selected).

short SetEndPeriod(BSTR *periodCode*);

Description:

Filters the end period of the selected product family

Parameters:

- *periodCode* - BSTR - The period code can be specified when a product family is selected to filter for period codes of contracts and series. When an empty string is passed, the end period code is cleared. The filter can be used in the following calculate and update methods:
  - CalcImplitedVolitility
  - CalcCallPutAverage
  - CalcPrice
  - CalcGreeks
  - CalcRiskArray
  - UpdatePrice
  - UpdateVolatility
  - UpdatePriceScan
  - UpdateVolScan
  - UpdateRiskFreeRate
  - UpdateDividendYield
  - UpdateTimeToExpiration

Remarks:

A product family must be selected. The end period code will be cleared when a blank string is passed, or if one of the following methods is called: ResetPeriods, SelectPIT, ResetPIT, Delete, SelectExchangeComplex, SelectBFCC, SelectProductFamily, ResetExchangeComplex, ResetBFCC, or ResetProductFamily

Return Value:

short - Returns 0 (success) or non-zero (failure - PF not selected).

void ResetPeriods();

Description:

Reset the start period and end period filters

Parameters:

None

Return Value:

None

short CopyPIT(BSTR *description*, BSTR *busDate*, short *isSettle*, short *isFinal*, BSTR *busTime*);

Description:

Use this method to copy Point In Time. This method creates new Point In Time object exactly matching the selected Point In Time. Attributes of the newly created Point In Time will be changed using parameters specified for the method. If Point In Time with these attributes already exists an error will be generated. Point In Time should be selected prior to using this method. If no Point In Time is selected an error will be generated. This method does not affect selected Point In Time.

Parameters:

- *description* - BSTR - The description of the point in time (can be blank)
- *busDate* - BSTR - The business date has following format: YYYYMMDD (blank – ignore)
- *isSettle* - short - Settlement indicator: 0 for non-settlement; 1 for settlement; or -1 for no preference on settlement indication.
- *isFinal* - short - Final flag: 0 for not final; 1 for final; or -1 for no preference.
- *busTime* - BSTR - The time of the point in time formatted as "HHMM" (can also be blank).

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

void DoMarketObservation();

Description:

Calculate a marked to market observation on the currently selected point in time.

Parameters:

None

Return Value:

None

void ResetMarketPrices();

Description:

Reset a marked to market observation on the selected point in time.

Parameters:

None

Return Value:

None

double GetPortfMTMFut();

Description:

Returns the marked to market future position on the selected portfolio.

Parameters:

None

Return Value:

double - the marked to market future position, or 0 if a portfolio is not selected

double GetPortfMTMPrem();

Description:

Returns the marked to market amount for premium style positions on the selected portfolio.

Parameters:

None

Return Value:

double - the marked to market option premium, or 0 if a portfolio is not selected

double GetPortfMTMTFut();

Description:

Returns the marked to market future style open trades on the selected portfolio.

Parameters:

None

Return Value:

double - the marked to market future trade, or 0 if a portfolio is not selected

double GetPortfMTMTPrem();

Description:

Returns the marked to market amount for premium style open trades on the selected portfolio.

Parameters:

None

Return Value:

double - the marked to market option premium trade, or 0 if a portfolio is not selected

double GetPortfUnrealizedPL();

Description:

Returns the unrealized P/L for the selected portfolio.

Parameters:

None

Return Value:

double - the unrealized P/L, or 0 if a portfolio is not selected

double GetPortfRealizedPLOpen();

Description:

Returns the realized P/L on open trades for the selected portfolio.

Parameters:

None

Return Value:

double - the realized open P/L, or 0 if a portfolio is not selected

double GetPortfRealizedPLLiq();

Description:

Returns the realized P/L for liquid dated trades on the selected portfolio.

Parameters:

None

Return Value:

double - the realized P/L liquidity, or 0 if a portfolio is not selected

short SaveRegistryTo(BSTR *fileName*);

Description:

Saves the trade registry to a specified location.

Parameters:

- *fileName* - BSTR - the name of the file to save the trade registry.

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short ApplyWhatIf(BSTR *fileName*, boolean *doCalc*);

Description:

Use this method to apply what-if scenario stored in the XML file *fileName* to the selected Point In Time. If Point In Time is not selected or file is not found, has incorrect format or is empty an error will be generated. Typical sequence of actions before using this command would include creation of a copy of a particular Point In Time. What-if scenario is applied to this copy:

- SelectPIT
- CopyPIT
- SelectPIT
- ApplyWhatIf

Parameters:

- *fileName* - BSTR - The fully specified path name of the what-if scenario XML file to load.
- *doCalc* - boolean - When set to true following actions are performed on the selected PointInTime after applying what-if scenario:
    - CalcPrice
    - CalcRiskArray
    - Calculate

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short CalcValues();

Description:

Calculate margin requirement. It works on the lowest level object already selected (portfolio or point in time). If point in time is not selected it will work on all loaded points in time.

Parameters:

None

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short CalcGreeks(short *meth*);

Description:

Calculate greeks. It works on the lowest level object already selected (product family, bfcc, exchange complex, point in time). If point in time is not selected it will work on all loaded points in time.

Parameters:

- *meth* - short - The calculation method code:

1	- Use option/series level volatility
2	- Use option level volatility only
3	- Use series level volatility only

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short ReplicateBasePrices();

Description:

Replicate base prices. It works on the lowest level object already selected (exchange complex or point in time). If neither an exchange complex nor a point in time is selected, the call will fail.

Parameters:

None

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short ApplyVolatilitySkew(short *meth*);

Description:

Skew volitility curve folowing a particular skew method. It works on the lowest level object already selected (exchange complex or point in time). If neither an exchange complex nor a point in time is selected, the call will fail.

Parameters:

- *meth* - short - The skew method code:

0	- Shifts the curve horizontally only
1	- Shifts the curve horizontally, vertical shift by Call/Put
2	- Shifts the curve horizontally, vertical shift by closest at the money or the average of Call/Put

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short PostTrade(BSTR *timeStamp*, long *tradeId*, BSTR *exchCmplxAcro*, BSTR *exchAcro*, BSTR *pfCode*, short *pfType*, BSTR *futPeriod*, BSTR *optPeriod*, boolean *isPut*, double *strike*, double *price*, long *tradeQty*);

Description:

Post a trade to the selected portfolio. If a portfolio isn't selected, the call will fail. After a trade is posted to the portfolio, margin will be recalculated.

Parameters:

- *timeStamp* - BSTR - The timestamp of the trade in a "YYYYMMDDHHMMSS" format.
- *tradeId* - long - The trade ID which will be written to the trade register.
- *exchCmplxAcro* - BSTR - The exchange complex acronym (“CME”, “BOTCC”, etc.)
- *exchAcro* - BSTR - The exchange acronym (“CME”, “CBT”, etc.)
- *pfCode* - BSTR - The product family code (“SP”, “BP”, etc.)
- *pfType* - short - The product family type. Possible values are:

1	- physical
2	- debt
3	- stock
11	- future
12	- equivalence debt
14	- forward
15	- interest rate swap
16	- E-Debt
17	- credit rate swap
21	- option on physical
22	- option on future
23	- option on equity
31	- combination
41	- option on combination

- *futPeriod* - BSTR - The period code for future contract (period code for the underlying contract for options). Format is usually YYYYMM, there are special cases like weeklies and flexes when some characters are added to this format (for example “200002W3”). For physicals, equity and options on physicals/equity this parameter should be set to “000000”.

- *optPeriod* - BSTR - The period code for option contract. This parameter and following contract identification parameters are ignored for non-option contracts. Same format as *futPeriod*.
- *isPut* - boolean - The put (true) or call (false) flag for options.
- *strike* - double - The strike price for options. For CDS products, this is the coupon.
- *price* - double - The price.
- *tradeQty* - long - The trade quantity.

Return Value:

short - Return value can be 0 (success) or non-zero (failure).

short ResetPrices(short *prodType*);

Description:

Reset prices for a given product type. It works on the lowest level object already selected (portfolio or point in time). If neither an exchange complex nor a point in time is selected, the call will fail.

Parameters:

- *prodType* - short - The product type to reset prices for. Possible values are:

-1	- All future based products
-2	- All Option based products
1	- physical
2	- debt
3	- stock
11	- future
12	- equivalence debt
14	- forward
15	- interest rate swap
16	- E-Debt
17	- credit rate swap
21	- option on physical
22	- option on future
23	- option on equity
31	- combination
41	- option on combination

Return Value:

short - Return value can be 0 (success) or non-zero (failure).