

CHAPTER 3: DATABASE ACCESS FUNCTIONS

ITEM OVERVIEW

CRSPAAccess items in the Stock & Indices databases and the CRSP\Compustat Merged database can be identified with unique mnemonic text items name maintained by CRSP, called itm_name. Items can be further qualified by a set of secondary keys. CRSP calls these known collections of keys and values a keyset and assigns a numeric code and mnemonic tag to each unique collection. Each of these represents different output series. When multiple keysets are available, the user can specify both the item and keyset to identify a series or use the default preset combination most commonly used.

For example, the Compustat data item SALE has secondary keys for industry format, data format, population source, and consolidation level. A different value of company sales can be available for any combination of these keys, such as a combination that represents the originally reported sales or the final restated sales from a later filing.

DATA ITEM GROUPING

- All itm_names are organized into groups for selection and presentation. Each group is given a name called grp_name. Grp_names are unique within the application and do not overlap with itm_name.

A group can include other groups or a group can include items. Items in a group must be compatible so that they can form a single table. For example, time series items must use the same calendar if using the same keyset. Items can belong to more than one group. Each group belongs to an entity class that describes the types of entities associated with items in the group. Company level and security level data are examples of entity classes.

Groups can be selected by grp_name, but designated groups can also be selected by a two-letter mnemonic code.

CRSP ITEM LIST SELECTION

CRSP item functions use a standard notation for specifying a set of data items. The notation allows selection by group or item. Examples are bal_ann or sale;at or prc;ret;vol or sale.*

For more, please see http://www.crsp.chicagobooth.edu/documentation/product/ccm/software/ccm_print/#data.

The item notation includes a high level item descriptor comprised of item elements, global qualifiers, and keyset specifications. The user provides information in one of three forms, with as many invocations as desired:

- **Full_list** - full description of items to select. It is in the form [global_section:]list_section
- **File+list** - description of items is in an input file and qualifiers
 - ♦ Specified in form [global_section:]filepath
 - ♦ Where file contains a list_element on each row.
- **Printopt** - a defined 2-letter shorthand for a defined or structure group. A printopt includes the 2-letter code optionally followed by keyset_list, separated by a period.

One list is maintained, and if an item/keyset is specified multiple times in the same handle it is ignored after the first one.

Selection components are:

- **Global_section** - optional section that modifies all elements in the list. It can contain the following markers:
 - ♦ **F** - add all children items with relation type F (footnotes)
 - ♦ **D** - add all children items with relation type D (data codes)
 - ♦ **K.keyset_list** - apply the keyset list to all items. See below for form of a keyset_list.

- `List_section` – semi-colon-delimited string of list elements: `list_element[;List_element...]`
- `List_element` – describes a CRSP item name or group name and keysets applied to it. The element name and keyset, if provided, are delimited by a period. A list element is in the form `elem_name[.keyset_list]`
- `Elem_name` – can be a CRSP item name or group name. Eventually all groups will be expanded to one or more items.
- `Keyset_list` – describes a set of keysets used for items. A keyset list is in the form
 - ♦ `*` - select all available keysets
 - ♦ `##,#...` - select all indicated keysets in a numeric list (examples 3 or 1-2 or 1,6-7,12 or 3-5,9-11)
 - ♦ `[empty]` – use the default keyset for all selected items.
- Calendar information including the data needed to identify periods for time series data.
- Key information for the current data loaded.
- Indexes directly to the items without the group context.

All item lists use a common `CRSP_ITM` structure for an item. It is identified by `itm_name` and `keyset` and contains pointers to the data objects plus pointers to relevant information in the reference arrays.

`CRSPAccess` provides a high level and flexible programming mechanism to support the large and changing set of supported data items. These item handling functions organize all items into groups (tables) based on an application identifier tied to a defined set of data using a standard notation and data structure. Users can specify any number of items with a standard mutation and traverse these items by group or look up individual fields. Item handling will work with any database with reference data. The `CRSPAccess` Programmers Guide contains functions for the CRSP Stock and Indices data.

ITEM HANDLE

All item handling activities for one application are stored in an item handle structure. This handle is passed as a parameter to most item handling functions. The structure contains the following types of information:

- A group table – an array of all groups available in the application. Each group includes information specific to that table –
 - ♦ An item list of all items and keysets selected for that table. All selected items are attached to one of the group tables.
 - ♦ A list of all keysets selected for that table
 - ♦ A list of structures and keysets if the group is a structure
- Set information, with database information including the set structure. The set structure is not accessed directly. The items are bound to the proper location in the set structure.
- Declare a `CRSP_ITM_HNDL` pointer for an `app_id` and initialize to `NULL`.
- Call the `crsp_itm_init` function to initialize the handle given the `app_id` and the database path. This will load all the reference data and open a shell of the database.
- Call the item load functions one or more times to select items of interest. `crsp_itm_load` should be sufficient for most usage. This will prepare the database and attach each item to its data location.
- There are five handle configuration flags that can be set to affect the behavior or the item handling. Each has a default value set during the initialization, but can be modified by setting a field in the handle.

ITEM FUNCTIONS

The structure of a program using the item handling functions takes the basic form:

- ♦ `keytype` - determines the keys used to select data in read functions. Supported keytypes for the application are included in the reference data. A default will be set.
 - ♦ `keyset_disp_cd` - determined whether keyset items are labeled by the keyset number (NUM), the keyset tag (TAG), or the expanded list of all items comprising the keyset (EXP). The default display is TAG.
 - ♦ `Fiscal_disp_cd` - determines whether fiscal-based time series items are reported on a calendar basis (C) or a fiscal basis (F). The default is C.
 - ♦ `Curr_disp_cd` - determines whether monetary values are reported in the currency reported by Compustat (REP) or in US Dollars (USD). The default currency display code is REP.
 - ♦ `Grp_fill_cd` - determines whether group item lists are filled so that every selected item is included for every selected keyset (Y or N). The default is Yes (Y).
- Call a function to load the key data into the handle and call `crsp_itm_get` as many times as needed to read desired data.
 - ♦ User can use `crsp_itm_set_key` to set individual input key items of interest or `crsp_itm_parse_key` to pass a string and have it parsed into one or more input key items.
 - ♦ The `crsp_itm_read` function will load all data according to items loaded and key information specified. The user can retrieve the value for any of the found keys after the read function with `crsp_itm_get_key`.
 - Processing of data is usually done after the appropriate `get` is executed and data are loaded. A user can process data by traversing the groups and the item lists, or use one of the item indexes created to find a specific item and process it. Since a `get` function can affect either master or detail data, the program must rely on the status to determine whether a class of data was changed by the `get` and

then process it accordingly.

- Call the `crsp_itm_close` function to close the handle and free its data.

ITEM USAGE

Normal data processing access is through an item structure named `CRSP_ITM`. `CRSP_ITM` includes metadata describing the item as well as the actual data retrieved from the database. The metadata is set once by `crsp_itm_open`, but the data that is loaded can change each time the `crsp_itm_read` function is called.

The item handling functions hide the internal storage and report the data according to `itm_name` and `keyset`, which can be found from among the loaded items with the item look-up function, `crsp_itm_find`.

CCM STRUCTURES

Selected Xpressfeed primary data groups and CRSP supplemental data groups are accessed by the entire group as a defined structure rather than as a stand-alone item. These data groups and their elements can both be accessed by `itm_name`, but recommended programming access is through the `itm_name` of the structure. To access the structure and its fields, load the structure `itm_name` during initialization, create a `CRSP_ITM` pointer matching the `itm_name` and attach it to the data, and access the structure and its fields through the pointer. The tables below show the data groups available as structures and their usage through the `CRSP_ITM` structure.

ITM_NAME	DESCRIPTION	C STRUCTURE NAME	OBJECT TYPE	OBJECT ACCESS VIA CRSP_ITM
MASTER	CCM company id and range data	CRSP_CCM_MASTER	row	master_itm->obj.row
COMPANY	CCM company header information	CRSP_CCM_COMPANY	row	company_itm->obj.row
IDX_INDEX	CCM idx_index header information	CRSP_CCM_IDX_INDEX	row	idx_index_itm->obj.row
SPIND	S&P index header (pre-GICS)	CRSP_CCM_SPIND	row	spind.itm->obj.row
COMPHIST	CCM company header history	CRSP_CCM_COMPHIST	array	comphist_itm->obj.arr
CSTHIST	CST header history	CRSP_CST_NAME	array	csthist_itm->obj.arr
LINK	link history	CRSP_CCM_LINK	array	link_itm->obj.arr
LINKUSED	CCM company CRSP link used data	CRSP_CCM_LINKUSED	array	linkused_itm->obj.arr
LINKRNG	CCM company CRSP link range data	CRSP_CCM_LINKRNG	array	linkused_itm->obj.arr
ADJFACT	CCM company adjustment factor history	CRSP_CCM_ADJFACT	array	adjfact.itm->obj.arr
HGIC	CCM company GICS code history	CRSP_CCM_HGIC	array	hgic_itm->obj.arr
OFFTITL	CCM company officer title data	CRSP_CCM_OFFTITL	array	offtitl_itm->obj.arr
CCM_FILEDATE	CCM company filing date data	CRSP_CCM_FILEDATE	array	ccm_filedate->obj.arr
CCM_IPCD	CCM industry presentation code data	CRSP_CCM_IPCD	array	ccm_filedate->obj.arr
SECURITY	CCM security header information	CRSP_CCM_SECURITY	row	security_itm->obj.row
SECHIST	CCM security header history	CRSP_CCM_SECHIST	array	sechist_itm->obj.arr
SEC_MTHSPT	CCM security monthly split events	CRSP_CCM_SEC_MTHSPT	row	sec_mthspt_itm->obj.row
SEC_MSPT_FN	CCM security monthly split event footnotes	CRSP_CCM_SEC_MTH_FN	row	sec_mspt_fn_itm->obj.row
SEC_MDIV_FN	CCM security monthly dividend event footnotes	CRSP_CCM_SEC_MTH_FN	row	sec_mddiv_fn_itm->obj.row
SEC_SPIND	CCM security S&P information events	CRSP_CCM_SEC_SPIND	row	sec_spind_itm->obj.row
IDXCST_HIS	CCM security historical index constituents	CRSP_CCM_IDXCST_HIS	array	idxcst_his_itm.obj.arr
SPIDX_CST	CCM security S&P index constituent events	CRSP_CCM_SPIDX_CST	array	spidx_cst_itm.obj.arr
CCM_SEGCUR	CCM opseg currency rate data	CRSP_CST_SEGCUR	array	ccm_segcur->obj.arr
CCM_SEGSRC	CCM opseg source data	CRSP_CST_SEGSRC	array	ccm_segsrc->obj.arr
CCM_SEGPROD	CCM opseg product data	CRSP_CST_SEGPROD	array	ccm_segprod->obj.arr
CCM_SEGCUST	CCM opseg customer data	CRSP_CST_SEGCUST	array	ccm_segcust->obj.arr
CCM_SEGDTL	CCM opseg detail data	CRSP_CST_SEGDTL	array	ccm_segdtl->obj.arr
CCM_SEGITM	CCM opseg item data	CRSP_CST_SEGITM	array	ccm_segitm->obj.arr
CCM_SEGNAICS	CCM opseg NAICS data	CRSP_CST_SEGNAICS	array	ccm_segnaics->obj.arr
CCM_SEGGEO	CCM opseg geographic data	CRSP_CST_SEGGEO	array	ccm_seggeo->obj.arr

CCM FIELD USAGE TABLE

STRUCTURE: MASTER

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
BEGQTR	Quarterly date of earliest data (yyyy.q)	int(4)	%6d	master_itm->arr.master->begqtr
BEGYR	Annual date of earliest data (yyyymmdd)	int(4)	%4d	master_itm->arr.master->begyr
CBEGDT	First date of Compustat data	int(4)	%8d	master_itm->arr.master->cbegdt
CCMID	Permanent record identifier for Compustat company or index data, represents GVKEY for company, GVKEYX for index	int(4)	%6d	master_itm->arr.master->ccmid
CCMIDTYPE	Type of key for Compustat data. 1 = company data, 2 = index data	int(4)	%2d	master_itm->arr.master->ccmidtype
CENDT	Last date of Compustat data	int(4)	%8d	master_itm->arr.master->cendt
ENDQTR	Quarterly date of last data (yyyy.q)	int(4)	%6d	master_itm->arr.master->endqtr
ENDYR	Annual date of last data (yyyymmdd)	int(4)	%4d	master_itm->arr.master->endyr

STRUCTURE: COMPANY

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
ADD1-4	Address lines 1-4	char(68)	%-65s	company_itm->arr.company->add#
ADDZIP	Postal code	char(24)	%-24s	company_itm->arr.company->addzip
BUSDESC	Business description	char(2000)	%-2000s	company_itm->arr.company->busdesc
CIK	CIK number	char(12)	%-10s	company_itm->arr.company->cik
CITY	City	char(104)	%-104s	company_itm->arr.company->city
CONM	Company name	char(256)	%-255s	company_itm->arr.company->conm
CONML	Company legal name	char(104)	%-100s	company_itm->arr.company->conml
COSTAT	Postal code	char(24)	%-1s	company_itm->arr.company->addzip
COUNTY	County code	char(104)	%-100s	company_itm->arr.company->county
DLDE	Research company deletion date	int(4)	%8d	company_itm->arr.company->dldte
DLRSN	Research company reason for deletion	char(12)	%-8s	company_itm->arr.company->dlrsn
EIN	Employer identification number	char(12)	%-10s	company_itm->arr.company->ein
FAX	Fax number	char(24)	%-18s	company_itm->arr.company->fax
FIC	ISO Country code of incorporation	char(16)	%-3s	company_itm->arr.company->fic
FYRC	Fiscal year end (current)	int(4)	%2d	company_itm->arr.company->fyrc
GGROUP	GICS groups	char(12)	%-4s	company_itm->arr.company->ggroup
GIND	GICS industries	char(12)	%-6s	company_itm->arr.company->gind
GSECTOR	GICS sectors	char(12)	%-2s	company_itm->arr.company->gsector
GSUBIND	GICS sub-industries	char(12)	%-8s	company_itm->arr.company->gsubind
IDBFLAG	International/Domestic/Both indicator	char(12)	%-1s	company_itm->arr.company->idbflag
INCORP	State/Province of incorporation code	char(12)	%-8s	company_itm->arr.company->incorp
IPODATE	Company initial public offering date	int(4)	%8d	company_itm->arr.company->ipodate
LOC	ISOCountry code/ headquarters	char(4)	%-3s	company_itm->arr.company->loc
NAICS	North American Industry Classification Code	char(8)	%-6s	company_itm->arr.company->naics
PHONE	Phone number	char(24)	%-18s	company_itm->arr.company->phone
PRICAN	Primary Issue Tag - Canada	char(12)	%-8s	company_itm->arr.company->prican
PRIROW	Primary Issue Tag – rest of world	char(12)	%-8s	company_itm->arr.company->prirow
PRIUSA	Primary Issue Tag - USA	char(12)	%-8s	company_itm->arr.company->priusa
SIC	SIC code	int(4)	%4d	company_itm->arr.company->sic
SPCINDCD	S&P industry sector code - reference	int(4)	%4d	company_itm->arr.company->spcindcd
SPCSECCD	S&P economic sector code - reference	int(4)	%4d	company_itm->arr.company->spcseccd
STATE	State/Province	char(12)	%-8s	company_itm->arr.company->state
STKO	Stock ownership code	int(4)	%1d	company_itm->arr.company->stko
WEBURL	Website address	char(68)	%-60s	company_itm->arr.company->weburl

STRUCTURE: IDX_INDEX

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
IDX13KEY	13 character key	char(16)	%-13s	idx_index_itm->arr.idx_index->idx13key
IDXCSTFLG	Index constituent flag	char(4)	%-2s	idx_index_itm->arr.idx_index->idxcstflg
INDEXCAT	Index category code	char(12)	%-10s	idx_index_itm->arr.idx_index->indexcat
INDEXGEO	Index geographical area	char(12)	%-10s	idx_index_itm->arr.idx_index->indexgeo
INDEXTYPE	Index type	char(12)	%-10s	idx_index_itm->arr.idx_index->indextype
INDEXVAL	Index value	char(12)	%-10s	idx_index_itm->arr.idx_index->indexval
SPII	S&P industry index code	int(4)	%4d	idx_index_itm->arr.idx_index->spii
SPMI	S&P major index code	int(4)	%4d	idx_index_itm->arr.idx_index->spmi
TICI	Issue trading ticker	char(12)	%-8s	idx_index_itm->arr.idx_index->tici
XCONM	Company Name (Index)	char(256)	%-255s	idx_index_itm->arr.idx_index->xconm
XINDEXID	Index ID	char(12)	%-12s	idx_index_itm->arr.idx_index->xindexid
XTIC	Ticker/trading symbol (index)	char(12)	%-12s	idx_index_itm->arr.idx_index->xtic

STRUCTURE: SPIND

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SPIID	S&P Industry ID	int(4)	%4d	spind_itm->arr.spind->spiid
SPIMID	S&P Major Index ID	int(4)	%4d	spind_itm->arr.spind->spimid
SPITIC	S&P Index ticker	char(12)	%-12s	spind_itm->arr.spind->spitic
SPIDESC	S&P Index industry description /reference	char(256)	%-256s	spind_itm->arr.spind->spidesc

STRUCTURE: COMPHIST

Comphist field usage assumes an initialized CRSP_ITM comphist_itm attached to the COMPHIST itm_name. Index I in field usage is between 0 and comphist_itm-> obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
HCHGDT	Comphist description effective date	int(4)	%8d	comphist_itm->arr.comphist[i].hchgdt
HCHGENDDT	Comphist description last effective date	int(4)	%8d	comphist_itm->arr.comphist[i].hchgenddt
HDLDE	Historical research company – deletion date	int(4)	%8d	comphist_itm->arr.comphist[i].hdlde
HFYRC	Historical fiscal year end month / current	int(4)	%10d	comphist_itm->arr.comphist[i].hfyrc
HIPODATE	Historical company official public offering date	int(4)	%10d	comphist_itm->arr.comphist[i].hipodate
HSIC	Historical SIC Code	int(4)	%10d	comphist_itm->arr.comphist[i].hsic
HSPCINDCD	Historical S&P Industry code	int(4)	%10d	comphist_itm->arr.comphist[i].hspcindcd
HSPCSECCD	Historical S&P Economic sector code	int(4)	%10d	comphist_itm->arr.comphist[i].hspcseccd
HSTKO	Historical stock ownership code	int(4)	%10d	comphist_itm->arr.comphist[i].hstko
HADD1...4	Historical address lines 1-4	char(68)	%-68s	comphist_itm->arr.comphist[i].hadd1#
HADDZIP	Historical postal code	char(68)	%-24s	comphist_itm->arr.comphist[i].haddzip
HBUSDESC	Historical business description	char(2000)	%-2000s	comphist_itm->arr.comphist[i].hbusdesc
HCIK	Historical CIK number	char(12)	%-12s	comphist_itm->arr.comphist[i].hcik
HCITY	Historical city	char(104)	%-104s	comphist_itm->arr.comphist[i].hcity
HCONM	Historical company name	char(256)	%-256s	comphist_itm->arr.comphist[i].hconm
HCONML	Historical legal company name	char(104)	%-104s	comphist_itm->arr.comphist[i].hconml

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
HCOSTAT	Historical active/inactive status marker	char(4)	%-4s	comphist_itm->arr.comphist[i].hcostat
HCCOUNTY	Historical county code	char(104)	%-104s	comphist_itm->arr.comphist[i].hccounty
HDLRSN	Historical research company reason for deletion	char(12)	%-12s	comphist_itm->arr.comphist[i].hdlrsn
HEIN	Historical employer identification number	char(12)	%-12s	comphist_itm->arr.comphist[i].hein
HFAX	Historical fax number	char(16)	%-16s	comphist_itm->arr.comphist[i].hfax
HFIC	Historical ISO country code / incorporation	char(16)	%-16s	comphist_itm->arr.comphist[i].hfic
HGGROUP	Historical GICS group	char(12)	%-12s	comphist_itm->arr.comphist[i].hgggroup
HGIND	Historical GICS industries	char(12)	%-12s	comphist_itm->arr.comphist[i].hgind
HGSECTOR	Historical GICS sector	char(12)	%-12s	comphist_itm->arr.comphist[i].hgsector
HGSUBIND	Historical GICS sub-industries	char(12)	%-12s	comphist_itm->arr.comphist[i].hgsubind
HIDBFLAG	Historical international, domestic, both indicator	char(12)	%-12s	comphist_itm->arr.comphist[i].hidbflag
HINCORP	Historical state/province of incorporation code	char(12)	%-12s	comphist_itm->arr.comphist[i].hincorp
HLOC	Historic ISO country code/ headquarters	char(4)	%-4s	comphist_itm->arr.comphist[i].hloc
HNAICS	Historical NAICS codes	char(8)	%-8s	comphist_itm->arr.comphist[i].hnaics
HPHONE	Historical phone number	char(16)	%-16s	comphist_itm->arr.comphist[i].hphone
HPRICAN	Historical primary issue tag - Canada	char(12)	%-12s	comphist_itm->arr.comphist[i].hprican
HPRIROW	Historical primary issue tag – rest of world	char(12)	%-12s	comphist_itm->arr.comphist[i].hprirow
HPRIUSA	Historical primary issue tag - US	char(12)	%-12s	comphist_itm->arr.comphist[i].hpriusa
HSTATE	Historical state/province	char(12)	%-12s	comphist_itm->arr.comphist[i].hstate
HWEBURL	Historical website url	char(68)	%-68s	comphist_itm->arr.comphist[i].hweburl

STRUCTURE - CSTHIST

Csthist field usage assumes an initialized CRSP_ITM csthist_itm attached to the CSTHIST itm_name. Index i in field usage is between 0 and csthist_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CST_CHGDT	CST History effective date	int(4)	%8d	csthist_itm->arr.csthist[i].cst_chgdt
CST_CHGENDDT	CST History last effective date	int(4)	%8d	csthist_itm->arr.csthist[i].cst_chgenddt
CST_DNUM	CST History industry code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_dnum
CST_FILE	CST History file identification code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_file
CST_ZLIST	CST History exchange listing and S&P Index code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_zlist
CST_STATE	CST History state identification code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_state
CST_COUNTY	CST History county identification code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_county
CST_STINC	CST History state incorporation code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_stinc
CST_FINC	CST History foreign incorporation code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_finc
CST_XREL	CST History industry index relative code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_xrel
CST_STK	CST History stock ownership code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_stk
CST_DUP	CST History duplicate file code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_dup
CST_CCNDX	CST History current Canadian index code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_ccndx
CST_GICS	CST History Global Industry Classification Standard Code	int(4)	%4d	csthist_itm->arr.csthist[i].cst_gics
CST_IPODT	CST History IPO date	int(4)	%4d	csthist_itm->arr.csthist[i].cst_ipodt
CST_FUNDF1	CST History fundamental file identification code 1	int(4)	%4d	csthist_itm->arr.csthist[i].cst_fund1
CST_FUNDF2	CST History fundamental file identification code 2	int(4)	%4d	csthist_itm->arr.csthist[i].cst_fundf2
CST_FUNDF3	CST History fundamental file identification code 3	int(4)	%4d	csthist_itm->arr.csthist[i].cst_fundf3

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CST_NAICS	CST History North American Industry Classification	char(8)	%-8s	csthist_itm->arr.csthist[i].cst_naics
CST_CPSPIN	CST History primary S&P Index marker	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_cpspin
CST_CSSPIN	CST History subset S&P Index marker	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_csspin
CST_CSSPII	CST History secondary S&P Index marker	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_csspii
CST_SUBDBT	CST History current S&P subordinated debt rating	char(8)	%-8s	csthist_itm->arr.csthist[i].cst_subdbt
CST_CPAPER	CST History current S&P commercial paper rating	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_cpaper
CST_SDBT	CST History current S&P senior debt rating	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_sdbt
CST_SDBTIM	CST History current S&P senior debt rating - footnote	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_sdbtim
CST_CNUM	CST History CUSIP issuer code	char(16)	%-16s	csthist_itm->arr.csthist[i].cst_cnum
CST_CIC	CST History issuer number	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_cic
CST_CONAME	CST History company name	char(64)	%-64s	csthist_itm->arr.csthist[i].cst_coname
CST_INAME	CST History industry name	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_iname
CST_SMBL	CST History stock ticker symbol	char(16)	%-16s	csthist_itm->arr.csthist[i].cst_smb1
CST_EIN	CST History employer identification number	char(16)	%-16s	csthist_itm->arr.csthist[i].cst_ein
CST_INCORP	CST History incorporation ISO country code	char(4)	%-4s	csthist_itm->arr.csthist[i].cst_incorp

STRUCTURE: LINK

Link field usage assumes an initialized CRSP_ITM link_itm attached to the LINK itm_name. Index i in field usage is between 0 and link_itm-> obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
LINKDT	Effective date of the link record	int(4)	%8d	link_itm->arr.link[i].linkdt
LINKENDDT	Last effective date of the link record	int(4)	%8d	link_itm->arr.link[i].linkenddt
LPERMNO	CRSP PERMNO link during link period	int(4)	%6d	link_itm->arr.link[i].lpermno
LPERMCO	CRSP PERMCO link during link period	int(4)	%10d	link_itm->arr.link[i].lpermco
LIID	Security identifier	char(4)	%-3s	link_itm->arr.link[i].liid
LNKTYPE	Link type code	char(4)	%-4s	link_itm->arr.link[i].lnktype
LINKPRIM	Primary security link marker	char(4)	%-1s	link_itm->arr.link[i].linkprim

STRUCTURE: LINKUSED

Linkused field usage assumes an initialized CRSP_ITM linkused_itm attached to the LINKUSED itm_name. Index i in field usage is between 0 and linkused_itm-> obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
ULINKDT	Effective date of the link	int(4)	%8d	linkused_itm->arr.linkused[i].ulinkdt
ULINKENDDT	Last effective date of the link	int(4)	%8d	linkused_itm->arr.linkused[i].ulinkenddt
ULINKID	Linkused row identifier	int(4)	%d	linkused_itm->arr.linkused[i].ulinkid
UGVKEY	GVKEY used in the link	int(4)	%6d	linkused_itm->arr.linkused[i].ugvkey
UPERMNO	CRSP PERMNO link during link period	int(4)	%6d	linkused_itm->arr.linkused[i].upermno
UPERMCO	CRSP PERMCO link during link period	int(4)	%6d	linkused_itm->arr.linkused[i].upermco
UIID	Used Security ID	char(4)	%-3s	linkused_itm->arr.linkused[i].uiid
USEDFLAG	Flag marking whether link is used in building composite record	char(4)	%d	linkused_itm->arr.linkused[i].usedflag
ULINKPRIM	Used link primary marker	char(4)	%-1s	linkused_itm->arr.linkused[i].ulinkprim
ULINKTYPE	Used link type	char(4)	%-4s	linkused_itm->arr.linkused[i].ulinktype

STRUCTURE: LINKRNG

Linkrng field usage assumes an initialized CRSP_ITM linkrng_itm attached to the LINKRNG itm_name. Index i in field usage is between 0 and linkrng_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
RLINKID	Linkused row identifier	int(4)	%8d	linkrng_itm->arr.linkrng[i].rlinkid
RKEYSET	Keyset applicable to range	int(4)	%8d	linkrng_itm->arr.linkrng[i].rkeyset
RCALID	Calendar applicable to range	int(4)	%8d	linkrng_itm->arr.linkrng[i].rcalid
RBEGIN	Beginning time series range of link	int(4)	%8d	linkrng_itm->arr.linkrng[i].rbegin
REND	Ending time series range of link	int(4)	%8d	linkrng_itm->arr.linkrng[i].rend
RPREVIND	Time series range immediately preceding the link	int(4)	%8d	linkrng_itm->arr.linkrng[i].rprevind
RBEGDT	Beginning calendar range of link	int(4)	%8d	linkrng_itm->arr.linkrng[i].rbegdt
RENDDT	Ending calendar range of link	int(4)	%8d	linkrng_itm->arr.linkrng[i].renddt
RPREVD	Ending calendar range preceding the link	int(4)	%8d	linkrng_itm->arr.linkrng[i].rprevd
RFISCAL_DATA_FLG	Type of time series, C-calendar or F-fiscal.	char(8)	%-8s	linkrng_itm->arr.linkrng[i].rfiscal_data_flg

STRUCTURE: ADJFACT

Adjfact field usage assumes an initialized CRSP_ITM adjfact_itm attached to the ADJFACT itm_name. Index i in field usage is between 0 and adjfact_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
EFFDATE	Effective date- company cumulative factor	int(4)	%10d	linkrng_itm->arr.linkrng[i].effdate
THRUDATE	Thu date – company cumulative factor	int(4)	%10d	linkrng_itm->arr.linkrng[i].thru
ADJEX	Cumulative adjustment factor by Ex-date	double(8)	%18.4f	adjfact_itm->arr.adjfact[i].adjex
ADJPAY	Cumulative adjustment factor by Pay-date	double(8)	%18.4f	adjfact_itm->arr.adjfact[i].adjpay

STRUCTURE: HGIC

Hgic field usage assumes an initialized CRSP_ITM hgic_itm attached to the HGIC itm_name. Index i in field usage is between 0 and hgic_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
INDFROM	Effective from (start) date	int(4)	%8d	hgic_itm->arr.hgic[i].indfrom
INDTHRU	Effective through (last) date	int(4)	%8d	hgic_itm->arr.hgic[i].indthru
GGROUPH	Industry group name	char(12)	%-12s	hgic_itm->arr.hgic[i].ggrouph
GINDH	Group industry	char(12)	%-12s	hgic_itm->arr.hgic[i].gindh
GSECTORH	Group industry sector	char(12)	%-12s	hgic_itm->arr.hgic[i].gsectorh
GSUBINDH	Group sub-industries	char(12)	%-12s	hgic_itm->arr.hgic[i].gsubindh

STRUCTURE: OFFTITL

Offtitl field usage assumes an initialized CRSP_ITM offtitl_itm attached to the OFFTITL itm_name. Index i in field usage is between 0 and offtitl_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
OFID	Officer ID	int(4)	%9d	offtitl_itm->arr.offtitl[i].ofid
OFCD	Officer title	char(16)	%-8s	offtitl_itm->arr.offtitl[i].ofcd
OFNM	Officer Name(s)	char(40)	%-39s	offtitl_itm->arr.offtitl[i].ofnm

STRUCTURE: CCM_FILEDATE

Ccm_filedate field usage assumes an initialized CRSP_ITM ccm_filedate_itm attached to the CCM_FILEDATE itm_name. Index i in field usage is between 0 and ccm_filedate_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
FDATE	Company filing data date	int(4)	%8d	ccm_filedate_itm->arr.ccm_filedate[i].fdate
FCONSOL	Company consolidation level filedate	char(2)	%-2s	ccm_filedate_itm->arr.ccm_filedate[i].fconsol
FPOPSRC	Population source filedate	char(2)	%-2s	ccm_filedate_itm->arr.ccm_filedate[i].fpopsrc
SRCTYPE	Document source type filedate	char(12)	%-12s	ccm_filedate_itm->arr.ccm_filedate[i].srctype
FILEDATE	Company filing date	int(4)	%8d	ccm_filedate_itm->arr.ccm_filedate[i].filedate

STRUCTURE: CCM_IPCD

Ccm_ipcd field usage assumes an initialized CRSP_ITM ccm_ipcd_itm attached to the CCM_IPCD itm_name. Index i in field usage is between 0 and ccm_ipcd_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
IPDATE	Industry presentation code data date	int(4)	%8d	ccm_ipcd_itm->arr.ccm_ipcd[i].ipdate
IPCONSOL	Level of consolidation (Industry presentation code)	char(2)	%-1s	ccm_ipcd_itm->arr.ccm_ipcd[i].ipconsol
IPPOPSRC	Population source (Industry presentation code)	char(2)	%-1s	ccm_ipcd_itm->arr.ccm_ipcd[i].ippopsrc
IPCD	Industry presentation code	char(2)	%-1s	ccm_ipcd_itm->arr.ccm_ipcd[i].ipcd

STRUCTURE: SECURITY

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
EXCHG	Stock exchange	int(4)	%4d	security_itm->arr.security->exchg
DLDEI	Security inactivation date	int(4)	%8d	security_itm->arr.security->dldei
IID_SEQ_NUM	IID sequence number	int(4)	%8d	security_itm->arr.security->iid_seq_num
SBEGDT	First date of Compustat data for issue	int(4)	%8d	security_itm->arr.security->sbegdt
SENDDT	Last date of Compustat data for issue	int(4)	%8d	security_itm->arr.security->senddt
IID	Issue ID	char(4)	%-3s	security_itm->arr.security->iid
SCUSIP	CUSIP	char(12)	%-12s	security_itm->arr.security->cusip
DLRSNI	Security inactivation code	char(12)	%-8s	security_itm->arr.security->dlrsni
DSCI	Security description	char(32)	%-28s	security_itm->arr.security->dsci
EPF	Earnings participation flag	char(4)	%-1s	security_itm->arr.security->epf

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
EXCNTRY	Stock exchange country code	char(4)	%-3s	security_itm->arr.security->excntry
ISIN	International security identification number	char(16)	%-12s	security_itm->arr.security->isin
SSECSTAT	Security status marker	char(4)	%-4s	security_itm->arr.security->ssecstat
SEDOL	SEDOL	char(8)	%-7s	security_itm->arr.security->sedol
TIC	Ticker/trading symbol	char(12)	%-8s	security_itm->arr.security->tic
TPCI	Issue type	char(12)	%-8s	security_itm->arr.security->tpci

STRUCTURE: SECHIST

Sechist field usage assumes an initialized CRSP_ITM sechist_itm attached to the SECHIST itm_name. Index i in field usage is between 0 and sechist_itm-> obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
HSCHGDT	Historical security change date	int(4)	%8d	sechist_itm->arr.sechist[i].hschgdt
HSCHGENDDT	Historical security change end date	int(4)	%8d	sechist_itm->arr.sechist[i].hschgenddt
HEXCHG	Historical stock exchange	int(4)	%10d	sechist_itm->arr.sechist[i].hexchg
HDLDEI	Historical security inactivation date	int(4)	%8d	sechist_itm->arr.sechist[i].hdldei
HIID_SEQ_NUM	Historical issue ID sequence number	int(4)	%10d	sechist_itm->arr.sechist[i].hiid_seq_num
HIID	Historical issue ID	char(4)	%-3s	sechist_itm->arr.sechist[i].hiid
HSCUSIP	Historical CUSIP	char(12)	%-12s	sechist_itm->arr.sechist[i].hscusip
HDLRSNI	Historical security inactivation code	char(12)	%-12s	sechist_itm->arr.sechist[i].hdhhrsni
HDSCI	Historical security description	char(32)	%-32s	sechist_itm->arr.sechist[i].hdsci
HEPF	Historical earnings participation flag	char(4)	%-4s	sechist_itm->arr.sechist[i].hepf
HEXCNTRY	Historical stock exchange country code	char(4)	%-4s	sechist_itm->arr.sechist[i].hexcntry
HISIN	Historical international security identification number	char(16)	%-16s	sechist_itm->arr.sechist[i].hisin
HSSECSTAT	Historical security status marker	char(4)	%-4s	sechist_itm->arr.sechist[i].hssecstat
HSEDOL	Historical SEDOL	char(8)	%-8s	sechist_itm->arr.sechist[i].hsedol
HTIC	Historical ticker/trading symbol	char(12)	%-12s	sechist_itm->arr.sechist[i].htic
HTPCI	Historical issue type	char(12)	%-12s	sechist_itm->arr.sechist[i].htpci

STRUCTURE: SEC_MTHSPT

Sec_mthspt field usage assumes an initialized CRSP_ITM sec_mthspt_itm attached to the MTHSPT itm_name. Index i in field usage is between 0 and mthspt_itm-> obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DATADATEM	Monthly adjustment factor data date	int(4)	%10d	sec_mthspt_itm->arr.sec_mthspt[i].datadatem
RAWPM	Raw adjustment factor – pay date - monthly	double(8)	%18.4f	sec_mthspt_itm->arr.sec_mthspt[i].rawpm
RAWXM	Raw adjustment factor – ex date - monthly	double(8)	%18.4f	sec_mthspt_itm->arr.sec_mthspt[i].rawxm

STRUCTURE: SEC_MSPT_FN

Sec_mspt_fn field usage assumes an initialized CRSP_ITM sec_smpt_fn_itm attached to the SEC_MSPT_FN itm_name. Index i in field usage is between 0 and sec_mspt_fn_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DATADATEMF	Monthly adjustment factor footnote data date	int(4)	%10d	sec_mspt_fn_itm->arr.sec_mspt_fn[i].datadatemf
DATAITEMMF	Monthly split footnote dataitem	char(8)	%-8s	sec_mspt_fn_itm->arr.sec_mspt_fn[i].dataitemmf
RAWPM_FN1..FN5	Raw adjustment factor – pay date – monthly – footnotes 1-5	char(4)	%-4s	sec_mspt_fn_itm->arr.sec_mspt_fn[i].rawpm_fn1..fn5
RAWXM_FN1..FN5	Raw adjustment factor – ex date – monthly – footnotes 1-5	char(4)	%-4s	sec_mspt_fn_itm->arr.sec_mspt_fn[i].rawxm_fn1..fn5

STRUCTURE: SEC_MDIV_FN

Sec_mdiv_fn field usage assumes an initialized CRSP_ITM sec_mdiv_fn_itm attached to the SEC_MDIV_FN itm_name. Index i in field usage is between 0 and sec_mdiv_fn_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DIVDATADATEMF	Monthly dividend footnote data date	int(4)	%10d	sec_mdiv_fn_itm->arr.sec_mdiv_fn[i].divdatadatemf
DIVDATAITEMMF	Monthly dividend footnote data item	char(8)	%-8s	sec_mdiv_fn_itm->arr.sec_mdiv_fn[i].divdataitemmf
DVPSPM_FN1..FN5	Dividend per share – pay date – monthly – footnotes 1-5	char(4)	%-4s	sec_mdiv_fn_itm->arr.sec_mdiv_fn[i].dvpspm_fn1..fn5
DVPSXM_FN1..FN5	Dividend per share – ex date – monthly – footnotes 1-5	char(4)	%-4s	sec_mdiv_fn_itm->arr.sec_mdiv_fn[i].dvpsxm_fn1..fn5

STRUCTURE: SEC_SPIND

Sec_spind field usage assumes an initialized CRSP_ITM sec_spind_itm attached to the SEC_SPIND itm_name. Index i in field usage is between 0 and sec_spind_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SPBEGBDATE	S&P Index event beginning date	int(4)	%10d	sec_spind_itm->arr.sec_spind[i].spbegdate
SPENDDATE	S&P Index event ending date	int(4)	%10d	sec_spind->arr.sec_spind[i].spenddate
SPHIID	S&P holdings industry index ID	int(4)	%4d	sec_spind_itm->arr.sec_spind[i].sphiid
SPHMID	S&P holdings major index ID	int(4)	%4d	sec_spind->arr.sec_spind[i].sphmid
SPHSEC	S&P holdings sector code	int(4)	%4d	sec_spind_itm->arr.sec_spind[i].sphsec
SPH100	S&P holdings S&P 100 marker	int(4)	%4d	sec_spind->arr.sec_spind[i].sph100
SPHCUSIP	S&P holdings CUSIP	char(12)	%-9s	sec_spind_itm->arr.sec_spind[i].sphcusip
SPHNAME	S&P holdings name	char(36)	%-31s	sec_spind->arr.sec_spind[i].sphname
SPHTIC	S&P holdings ticker	char(12)	%-8s	sec_spind_itm->arr.sec_spind[i].sphitic
SPHVG	S&P holdings value/growth indicator	char(4)	%-1s	sec_spind->arr.sec_spind[i].sphvg

STRUCTURE: IDX CST HIS

Idxcst_his field usage assumes an initialized CRSP_ITM idx_csthis_itm attached to the IDX_CSTHIS itm_name. Index i in field usage is between 0 and idx_csthis_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
XFROM	S&P constituent from event date	int(4)	%10d	idx_csthis_itm->arr.idx_csthis[i].xfrom
IDX13KEY	S&P 13 character key	char(16)	%-13s	Idx_csthis->arr.idx_csthis[i].idx13key
XGVKETX	S&P constituent event index GVKEY	int(4)	%10d	Idx_csthisitm->arr.idx_csthis[i].xgvkeyx

STRUCTURE: SPIDX CST

Spidx_cst field usage assumes an initialized CRSP_ITM spidx_cst_itm attached to the SPIDX_CST itm_name. Index i in field usage is between 0 and spidx_cst_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SXBEGDATE	S&P constituent event beginning date	int(4)	%10d	spidx_cst_itm->arr.spidx_cst[i].sxbegdate
SXENDDATE	S&P constituent event ending date	int(4)	%10d	spidx_cst_itm->arr.spidx_cst[i].sxenddate
SPFLOAT	S&P float shares	int(4)	%4d	spidx_cst_itm->arr.spidx_cst[i].spfloat
INDEXID	S&P major index ID	char(12)	%-10s	spidx_cst_itm->arr.spidx_cst[i].indexid
EXCHGX	S&P constituent exchange	char(8)	%-4s	spidx_cst_itm->arr.spidx_cst[i].exchg
TICX	S&P holdings ticker	char(12)	%-10s	spidx_cst_itm->arr.spidx_cst[i].ticx
CUSIPX	S&P constituent CUSIP	char(12)	%-9s	spidx_cst_itm->arr.spidx_cst[i].cusipx
CONMX	S&P constituent name	char(44)	%-40s	spidx_cst_itm->arr.spidx_cst[i].conmx
CONTYPE	S&P constituent type	char(12)	%-10s	spidx_cst_itm->arr.spidx_cst[i].contype
CONVAL	S&P constituent value	char(12)	%-10s	spidx_cst_itm->arr.spidx_cst[i].conval

STRUCTURE: CCM_SEG CUR

Ccm_segcur field usage assumes an initialized CRSP_ITM ccm_segcur_itm attached to the CCM_SEG CUR itm_name. Index i in field usage is between 0 and ccm_segcur_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SC_DATAYR	Data year	int(4)	%4d	ccm_segcur_itm->arr.ccm_segcur[i].sc_datayr
SC_DATAFYR	Data fiscal year end month	int(4)	%2d	ccm_segcur_itm->arr.ccm_segcur[i].sc_datafyr
SC_CALYR	Data calendar year	int(4)	%4d	ccm_segcur_itm->arr.ccm_segcur[i].sc_calyr
SC_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_segcur_itm->arr.ccm_segcur[i].sc_srcfyr
SC_XRATE	Period end exchange rate	double(8)	%16.8f	ccm_segcur_itm->arr.ccm_segcur[i].sc_xrate
SC_XRATE12	12-month moving exchange rate	double(8)	%16.8f	ccm_segcur_itm->arr.ccm_segcur[i].sc_xrate12
SC_SRCCUR	Source currency code	char(4)	%-3s	ccm_segcur_itm->arr.ccm_segcur[i].sc_srccur
SC_CURCD	ISO currency code (USD)	char(4)	%-3s	ccm_segcur_itm->arr.ccm_segcur[i].sc_curcd

STRUCTURE: CCM_SEGSRC

Ccm_segsrc field usage assumes an initialized CRSP_ITM ccm_segsrc_itm attached to the CCM_SEGSRC itm_name. Index i in field usage is between 0 and ccm_segsrc_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SS_SRCYR	Source year	int(4)	%4d	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_srcyr
SS_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_srcfyr
SS_CALYR	Data calendar year	int(4)	%4d	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_calyr
SS_RCST1	Reserved 1	int(4)	%4d	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_rcst1
SS_SSRCE	Source document code	char(4)	%-2s	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_ssrce
SS_SUCODE	Source update code	char(4)	%-2s	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_sucode
SS_CURCD	ISO currency code	char(4)	%-3s	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_curcd
SS_SRCCUR	Source ISO currency code	char(4)	%-3s	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_srccur
SS_HNAICS	Segment primary historical NAICS	char(8)	%-6s	ccm_segsrc_itm->arr.ccm_segsrc[i].ss_hnaics

STRUCTURE: CCM_SEGPROD

Ccm_segprod field usage assumes an initialized CRSP_ITM ccm_segprod_itm attached to the CCM_SEGPROD itm_name. Index i in field usage is between 0 and ccm_segprod_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SP_SRCYR	Source year	int(4)	%4d	ccm_segprod_itm->arr.ccm_segprod[i].sp_srcyr
SP_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_segprod_itm->arr.ccm_segprod[i].sp_srcfyr
SP_PDID	Product identifier	int(4)	%4d	ccm_segprod_itm->arr.ccm_segprod[i].sp_pdid
SP_PSID	Segment link – segment identifier	int(4)	%4d	ccm_segprod_itm->arr.ccm_segprod[i].sp_psid
SP_PSALE	External revenues	float(4)	%10.4f	ccm_segprod_itm->arr.ccm_segprod[i].sp_psale
SP_RCST1	Reserved 1	float(4)	%10.4f	ccm_segprod_itm->arr.ccm_segprod[i].sp_rcst1
SP_PNAICS	Product NAICS code	char(8)	%-6s	ccm_segprod_itm->arr.ccm_segprod[i].sp_pnaics
SP_PSTYPE	Segment link- segment type	char(16)	%-83s	ccm_segprod_itm->arr.ccm_segprod[i].sp_pstype
SP_PNAME	Product name	char(64)	%-64.64s	ccm_segprod_itm->arr.ccm_segprod[i].sp_pname

STRUCTURE: CCM_SEGCUST

Ccm_segcust field usage assumes an initialized CRSP_ITM ccm_segcust_itm attached to the CCM_SEGCUST itm_name. Index i in field usage is between 0 and ccm_segcust_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SC_SRCYR	Source year	int(4)	%4d	ccm_segcust_itm->arr.ccm_segcust[i].sc_srcyr
SC_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_segcust_itm->arr.ccm_segcust[i].sc_srcfyr
SC_CDID	customer identifier	int(4)	%4d	ccm_segcust_itm->arr.ccm_segcust[i].sc_cdid
SC_CSID	Segment link – segment identifier	int(4)	%4d	ccm_segcust_itm->arr.ccm_segcust[i].sc_csid
SC_CSALE	customer revenues	float(4)	%10.4f	ccm_segcust_itm->arr.ccm_segcust[i].sc_csale
SC_RCST1	Reserved 1	int(4)	%4d	ccm_segcust_itm->arr.ccm_segcust[i].sc_rcst1
SC_CTYPE	Customer type	char(16)	%-8s	ccm_segcust_itm->arr.ccm_segcust[i].sc_ctype
SC_CGEOCD	Geographic area code	char(16)	%-8s	ccm_segcust_itm->arr.ccm_segcust[i].sc_cgeocd
SC_CGEOAR	Geographic area type	char(16)	%-8s	ccm_segcust_itm->arr.ccm_segcust[i].sc_cgeoar
SC_CSTYPE	Segment link – segment type	char(16)	%-8s	ccm_segcust_itm->arr.ccm_segcust[i].sc_cstype
SC_CNAME	Customer name data	char(64)	%-64.64s	ccm_segcust_itm->arr.ccm_segcust[i].sc_cname

STRUCTURE: CCM_SEGDTL

Ccm_segdtl field usage assumes an initialized CRSP_ITM ccm_segdtl_itm attached to the CCM_SEGDTL itm_name. Index i in field usage is between 0 and ccm_segdtl_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SD_SRCYR	Source year	int(4)	%4d	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_srcyr
SD_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_srcfyr
SD_SID	Segment identifier	int(4)	%4d	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_sid
SD_RCST1	Reserved 1	int(4)	%4d	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_rcst1
SD_STYPE	Segment type	char(16)	%-8s	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_stype
SD_SOPTP1	Operating segment type 1	char(16)	%-8s	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_stype
SD_SOPTP2	Operating segment type	char(16)	%-8s	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_cgeocd
SD_SGEOTP	Geographic segment type	char(16)	%-8s	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_cgeoar
SD_SNAME	Segment name	char(256)	%-64.64s	ccm_segdtl_itm->arr.ccm_segdtl[i].sd_cname

STRUCTURE: CCM_SEGITM

Ccm_segitm field usage assumes an initialized CRSP_ITM ccm_segitm_itm attached to the CCM_SEGITM itm_name. Index i in field usage is between 0 and ccm_segitm_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SI_DATYR	Data year	int(4)	%4d	ccm_segitm_itm->arr.ccm_segitm[i].si_datyr
SI_FISCYR	Data fiscal year end month	int(4)	%4d	ccm_segitm_itm->arr.ccm_segitm[i].si_fiscyr
SI_SRCYR	Source year	int(4)	%4d	ccm_segitm_itm->arr.ccm_segitm[i].si_srcyr
SI_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_segitm_itm->arr.ccm_segitm[i].si_srcfyr
SI_CALYR	Data calendar year	int(4)	%4d	ccm_segitm_itm->arr.ccm_segitm[i].si_calyr
SI_SID	Segment identifier	int(4)	%4d	ccm_segitm_itm->arr.ccm_segitm[i].si_sid
SI_EMP	Employees	int(4)	%9d	ccm_segitm_itm->arr.ccm_segitm[i].si_emp
SI_SALE	Net sales	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_sale
SI_OIBD	Operating income before depreciations	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_oibd
SI_DP	Depreciation & amortization	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_dp
SI_OIAD	Operating income after depreciation	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_oiad
SI_CAPX	Capital expenditures	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_capx
SI_IAT	Identifiable assets	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_iat
SI_EQEARN	Equity in earnings	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_eqearn
SI_INVEQ	Investments at equity	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_inveq
SI_RD	Research & development	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_rd
SI_OBKLG	Order backlog	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_obklg
SI_EXPORTS	Export sales	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_exports
SI_INTSEG	Inter-segment eliminations	int(4)	%4d	ccm_segitm_itm->arr.ccm_segitm[i].si_intseg
SI_OPINC	Operating profit	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_opinc
SI_PI	Pretax income	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_pi
SI_IB	Income before extraordinary earnings	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_ib
SI_NI	Net income (loss)	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_ni
SI_RCST1	Reserved 1	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_rcst1
SI_RCST2	Reserved 2	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_rcst2

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SI_RCST3	Reserved 3	float(4)	%10.4f	ccm_segitm_itm->arr.ccm_segitm[i].si_rcst3
SI_SALEF	Footnote 1 - sales	char(16)	%-8s	ccm_segitm_itm->arr.ccm_segitm[i].si_salef
SI_OPINCF	Footnote 2 – operating profit	char(16)	%-8s	ccm_segitm_itm->arr.ccm_segitm[i].si_opincf
SI_CAPXF	Footnote 3 – capital expenditures	char(16)	%-8s	ccm_segitm_itm->arr.ccm_segitm[i].si_capxf
SI_EQEARNF	Footnote 4 – equity in earnings	char(16)	%-8s	ccm_segitm_itm->arr.ccm_segitm[i].si_eqearnf
SI_EMPF	Footnote 5 - employees	char(16)	%-8s	ccm_segitm_itm->arr.ccm_segitm[i].si_empf
SI_RDF	Footnote 6 – research & development	char(16)	%-8s	ccm_segitm_itm->arr.ccm_segitm[i].si_rdf
SI_STYPE	Segment type	char(16)	%-8s	ccm_segitm_itm->arr.ccm_segitm[i].si_stype

STRUCTURE: CCM_SEGNAICS

Ccm_segnaics field usage assumes an initialized CRSP_ITM ccm_segnaics_itm attached to the CCM_SEGNAICS itm_name. Index i in field usage is between 0 and ccm_segnaics_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SN_SRCYR	Source year	int(4)	%4d	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_srcyr
SN_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_srcfyr
SN_SID	Segment identifier	int(4)	%4d	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_sid
SN_RCST1	Reserved 1	int(4)	%4d	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_rcst1
SN_STYPE	Segment type	char(16)	%-8s	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_stype
SN_SNAICS	NAICS code	char(8)	%-6s	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_snaics
SN_RANK	Ranking	int(4)	%4d	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_rank
SN_SIC	Segment SIC code	int(4)	%4d	ccm_segnaics_itm->arr.ccm_segnaics[i].sn_sic

STRUCTURE: CCM_SEGCEO

Ccm_seggeo field usage assumes an initialized CRSP_ITM ccm_seggeo_itm attached to the CCM_SEGCEO itm_name. Index i in field usage is between 0 and ccm_seggeo_itm->obj.arr->num -1.

MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SG_SRCYR	Source year	int(4)	%4d	ccm_seggeo_itm->arr.ccm_seggeo[i].sg_srcyr
SG_SRCFYR	Source fiscal year end month	int(4)	%2d	ccm_seggeo_itm->arr.ccm_seggeo[i].sg_srcfyr
SG_SID	Segment identifier	int(4)	%4d	ccm_seggeo_itm->arr.ccm_seggeo[i].sg_sid
SG_RCST1	Reserved 1	int(4)	%4d	ccm_seggeo_itm->arr.ccm_seggeo[i].sg_rcst1
SG_STYPE	Segment type	char(16)	%-8s	ccm_seggeo_itm->arr.ccm_seggeo[i].sg_stype
SG_SGEOCD	Geographic area code	char(16)	%-8s	ccm_seggeo_itm->arr.ccm_seggeo[i].sg_sgeocd
SG_SGEOTP	Geographic area type	char(16)	%-8s	ccm_seggeo_itm->arr.ccm_seggeo[i].sg_sgeotp

ITEM ACCESS FUNCTIONS

crsp_itm_init

Prepare a handle for item handling operations

PROTOTYPE:	<code>int crsp_itm_init(CRSP_ITM_HNDL **hndl, char *dbpath, int app_id, char *hndl_name)</code>
DESCRIPTION:	Prepare a handle for item handling operation for one database and one application id. The handle will be initialized and the database set type and set id identified, allowing loading of reference data and allocation of a set structure.
ARGUMENTS:	<p><code>CRSP_ITM_HNDL **hndl</code> - Double pointer to the item handle that will be used to manage the database information and item lists.</p> <p><code>char *dbpath</code> - Pointer to the database containing the data to load and the applicable reference data.</p> <p><code>int app_id</code> - Identifier of a defined application organizing data items into groups for access. Available <code>app_ids</code> can be found in the reference array, <code>crsp_itm_app</code>. Common <code>app_ids</code> have defined constants:</p> <ul style="list-style-type: none"> <code>CRSP_CCMITEMS_ID = 7</code> (generic CCM usage application) <p><code>char *hndl name</code> - Name to assign to the handle.</p>
RETURN VALUES:	<p><code>CRSP_SUCCESS:</code> If initialized successfully</p> <p><code>CRSP_FAIL:</code> If there is an error in the parameter, database cannot be opened, reference data unavailable, incompatibility between database and <code>app_id</code>.</p>
SIDE EFFECTS:	<p>If successful, the handle data are loaded:</p> <ul style="list-style-type: none"> The handle itself is allocated if not already allocated. The handle fields are initialized, including all lists and arrays. The <code>ca_ref</code> structure is loaded with the reference data in the database. If an old database with no reference data, it will use a global reference file with a standard name based on the <code>app_id</code> in the <code>CRSP_LIB</code> directory. <code>itm_grp</code> and <code>itm_avail</code> arrays in the handle are loaded with available tables and items <code>Set_list</code> element is allocated using the database path and <code>setid</code>. The database is opened with a 0 wanted, which loads reference data but allocates no module space. The root information is loaded to the set's <code>CRSP_ROOT_INFO</code> structure.
PRECONDITIONS	The item handle must be initialized to NULL or point to an allocated handle that can be re-initialized. The <code>app_id</code> must exist in the reference data of the database opened.

crsp_itm_open

Opens databases indicated by a handle and registers selected items.

PROTOTYPE:	<code>int crsp_itm_open (CRSP_ITM_HNDL **hndl)</code>
DESCRIPTION:	Registers selected items in a handle by expanding structures and keysets, preparing keys, determining modules needed to access items, opens the needed modules, and binds data in the item lists to the data structure locations. It also builds a master index of all items available in the handle.
ARGUMENTS:	<code>CRSP_ITM_HNDL **hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list.
RETURN VALUES:	SUCCESS: If opens successfully and binds the data
	CRSP_FAIL: If error in parameters, inconsistent handle, error opening databases or binding items.
SIDE EFFECTS:	If successful, the handle is ready for access: <ul style="list-style-type: none"> • The <code>itm_set</code> element for the database will be updated with the wanted needed. • Prepare supplementary lists for table keyset and calendar items, and handle key items. • The <code>itm_set</code> database will be opened with the needed wanted. • The <code>itm_idx</code> master list of items will be built from selected items. • All items in the list will have object pointer set to the data location in the set data structure. • If the handle <code>grp_fill_cd</code> is "Y", then the item lists are filled to ensure full tables. Filling creates items to ensure that every <code>itm_name</code> and keyset present in a group each combination is present even if not specified. Filling also arranges the lists so if multiple keysets, each is sorted in the same order as the first keyset seen.
PRECONDITIONS	The item handle must be previously initialized with <code>crsp_itm_init</code> . It generally follows one or more instances of item load functions.

crsp_itm_clear

Sets missing values in objects for all items in the handle.

PROTOTYPE:	<code>int crsp_itm_clear (CRSP_ITM_HNDL *hndl, int clear_flag)</code>
DESCRIPTION:	Sets missing values in the objects for all items in a handle. A flag determines how the missing values are set.
ARGUMENTS:	<code>CRSP_ITM_HNDL *hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list. <code>int clear_flag</code> - Code determining how the object will be cleared. Possible values are: <ul style="list-style-type: none"> • <code>CRSP_CLEAR_INIT</code> - only reset num for <code>CRSP_ARRAYS</code> and beg and end for <code>CRSP_TIMESERIES</code>, nothing for <code>CRSP_ROWS</code>. • <code>CRSP_CLEAR_LOADED</code> - only set missing values in a time series or array if non-empty. • <code>CRSP_CLEAR_RANGE</code> - set missing values for elements between beg and end for <code>CRSP_TIMESERIES</code>, between 0 and num - 1 for <code>CRSP_ARRAYS</code>, and all for <code>CRSP_ROWS</code>. • <code>CRSP_CLEAR_ALL</code> - set range to missing and set missing values for all elements in the object arrays. • <code>CRSP_CLEAR_SET</code> - place missing values in the 0th element of a <code>CRSP_TIMESERIES</code> array or the maxarr-1th element of a <code>CRSP_ARRAY</code> to missing values specific to the array type, or all missing values in a <code>CRSP_ROW</code> element.
RETURN VALUES:	CRSP_SUCCESS : If data loaded successfully CRSP_FAIL : If error with parameters, inconsistent handle, or unknown object type, array type, or all missing values in a CRSP ROW element.
SIDE EFFECTS:	Object pointers found in the handle will be cleared based on the <code>clear_flag</code> .
PRECONDITIONS	The item handle must be previously opened and objects bound with <code>crsp_itm_open</code> .

crsp_itm_load_key

Sets the key to be used for reads.

PROTOTYPE:	<code>int crsp_itm_load_key(CRSP_ITM_HNDL *hndl, char *keytype)</code>
DESCRIPTION:	Defines the keytype that will be used for subsequent reads.
ARGUMENTS:	<p><code>CRSP_ITM_HNDL *hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>char * keytype</code> - Name of the key to initialize. Values are:</p> <ul style="list-style-type: none"> • <code>gvkey</code> – Compustat company key (default) • <code>gvkeyx</code> – Compustat index key • <code>ccmid</code> – <code>gvkey</code> or <code>gvkeyx</code> • <code>permno</code> – CRSP permno found in any links • <code>permco</code> – CRSP permco found in any links • <code>apermno</code> – CRSP-centric composite records by permno • <code>ppermco</code> – CRSP-centric composite records by permno – primary links only • <code>sic</code> – Compustat company SIC code • <code>ticker</code> – Compustat security ticker symbol • <code>cusip</code> – security CUSIP
RETURN VALUES:	<p><code>CRSP_SUCCESS</code> : If successful</p> <p><code>CRSP_FAIL</code> : Error in parameters, handle not initialized, or keytype not found.</p>
SIDE EFFECTS:	If successful, the handle is prepared to handle reads.
PRECONDITIONS:	The item handle must be initialized. Keytype must be known for the <code>app_id</code> .

crsp_itm_set_key

Sets the key specifications to be used with selecting data.

PROTOTYPE:	<code>int crsp_itm_set_key (CRSP_ITM_HNDL *hndl, char *key_itm, void *keyval)</code>
DESCRIPTION:	Loads key information that will be used to load data in a <code>crsp_itm_read</code> call. The key is setup during the <code>crsp_itm_open</code> based on the active keytype. The value passed to this function and entered into the handle attached to the input key item.
ARGUMENTS:	<p><code>CRSP_ITM_HNDL *hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>Char *key_itm</code> - String containing an <code>itm_name</code> of an input key item to be loaded.</p> <p><code>Void keyval</code> - Data to be loaded into the key item. Data must agree with the item's type.</p>
RETURN VALUES:	<p><code>CRSP_SUCCESS</code> : If data loaded successfully</p> <p><code>CRSP_FAIL</code> : If error in parameters, handle not open, key item.</p>
SIDE EFFECTS:	If successful, the <code>keyval</code> is copied into the data location for the input key item element in the handle.
PRECONDITIONS:	The item handle must be initialized and opened. The item key array must be initialized based on a keytype with the <code>crsp_itm_open</code> or <code>crsp_itm_init_key</code> functions. The <code>key_itm</code> must be a valid item for that keytype, and the <code>keyval</code> data must agree with the type of that item.

crsp_itm_get_key

Gets the key found by `crsp_itm_read` for a named key item.

PROTOTYPE:	<code>int crsp_itm_get_key (CRSP_ITM_HNDL *hndl, char *key_itm, void *keyval)</code>
DESCRIPTION:	Retrieves key information for data loaded by a <code>crsp_itm_read</code> call. An output key item list is prepared when the key is initialized, and loaded by <code>crsp_itm_read</code> . This function finds the <code>key_itm</code> in the list and copies the value into the user's location.
ARGUMENTS:	<p><code>CRSP_ITM_HNDL *hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>Char *key_itm</code> - String containing an <code>itm_name</code> of a loaded key to be retrieved.</p> <p><code>Void *keyval</code> - Location to place the key value. Location must agree with the item's type and size.</p>
RETURN VALUES:	<p><code>CRSP_SUCCESS</code>: If data loaded successfully</p> <p><code>CRSP_FAIL</code>: If error in parameters, handle not open, key item.</p>
SIDE EFFECTS:	If successful, the <code>keyval</code> is loaded based on the item and key value type.
PRECONDITIONS:	The item handle must be initialized and opened. The item key array must be initialized based on a keytype with the <code>crsp_itm_open</code> or <code>crsp_itm_init_key</code> functions. The <code>key_itm</code> must be a valid item for that keytype, and the <code>keyval</code> data must agree with the type of that item.

crsp_itm_parse_key

Sets the key specifications to be used when selecting data based on a text string.

PROTOTYPE:	<code>int crsp_itm_parse_key (CRSP_ITM_HNDL *hndl, char *key_text)</code>
DESCRIPTION:	<p>Loads key information in text format that will be used to load data in a <code>crsp_itm_read</code> call. The key is setup during <code>crsp_itm_open</code> based on the keytype identifier. The <code>key_text</code> string is parsed in the prescribed mapping order of the keytype and loaded into the handle.</p> <p>The <code>key_text</code> is in the form <code>key1.key2...</code>, where input key items are separated by periods. If an input key is not provided it will be set with a missing value. For example, if the keytype is <code>gvkey</code>, to access IBM company and security data of its primary security, <code>key_text</code> will be set to "6066.01". IID is optional if only company items are selected. In this case, "6066" may be used.</p>
ARGUMENTS:	<p><code>CRSP_ITM_HNDL *hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>Void *key_text</code> - String containing key information on interest, in order of keys, with each item separated by a period.</p>
RETURN VALUES:	<p><code>CRSP_SUCCESS</code>: If successfully loaded.</p> <p><code>CRSP_FAIL</code>: If bad parameter, handle not open, key item.</p>
SIDE EFFECTS:	If successful, the values are copied into the handle input key item data locations for each input key item from the <code>key_text</code> string.
PRECONDITIONS:	The item handle must be initialized and opened. The item key array must be initialized based on a keytype with the <code>crsp_itm_open</code> or <code>crsp_itm_init_key</code> functions.

crsp_itm_read

Loads data into a handle based on provided keys, supporting possible secondary indexes.

PROTOTYPE:	<code>int crsp_itm_read (CRSP_ITM_HNDL *hndl, int keyflag, int *status)</code>
DESCRIPTION:	<p>Loads data from handle based on item keys specified in prior <code>crsp_itm_key</code> calls and <code>keyflag</code>. Depending on the level of the entity class, the operation may include reading data from the database into structures and/or specifying data already loaded. This allows a direct or positional read based on <code>keyflag</code>.</p> <p>If the handle <code>fiscal_disp_cd</code> is C, any fiscal-based time series are shifted to a calendar basis as part of the read operation.</p>
ARGUMENTS:	<p><code>CRSP_ITM_HNDL *hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>int keyflag</code> - Code determining how the key is interpreted. <code>CRSP_EXACT</code> to look for a specific value, <code>CRSP_BACK</code> or <code>CRSP_FORWARD</code> for direct selection when partial matches are allowed, or a positional qualifier to base selection on the position relative to the last key accessed. Qualifiers include:</p> <ul style="list-style-type: none"> • <code>CRSP_NEXT</code> (<code>--99</code>) - read next key in sequence • <code>CRSP_PREV</code> (<code>--96</code>) - read previous key in sequence • <code>CRSP_SAME</code> (<code>--98</code>) - read same key, possibly with different information • <code>CRSP_FIRST</code> (<code>--95</code>) - read first key in the database • <code>CRSP_LAST</code> (<code>--97</code>) - read last key in the database <p><code>Int *status</code> - User provided location to load with the level of the read. It will be loaded with a 0 if the load results in reading new master data. It will be loaded with a number greater than 0 if the load impacts detail or global data, but no master data are affected.</p>
RETURN VALUES:	<p><code>CRSP_SUCCESS</code>: If data loaded successfully</p> <p><code>CRSP_EOF</code>: If positional read reaches the end of the file</p> <p><code>CRSP_NOT_FOUND</code>: If key not found on exact read. If a detail input key is not provided and no items of that entity class are selected, the return is <code>CRSP_SUCCESS</code> as long as the primary key matches.</p> <p><code>CRSP_FAIL</code>: If error in parameters, handle not opened, error in read operations.</p>
SIDE EFFECTS:	If successful, the wanted data for the key are loaded into the handle set structure which allows item objects to point to the loaded data. The key found for each level is loaded into the outkey item list. If the handle <code>fiscal_disp_cd</code> is set to calendar-based and items are fiscal-based, shifted calendars are created and time series are converted to calendar basis. The status argument is loaded based on whether the primary key changed. Handle <code>primkey</code> field and <code>readlvl</code> are set. <code>Readlvl</code> is set to the rank of the first entity class changed. If the primary key changed, <code>getlvl</code> is set to 0.
PRECONDITIONS	The item handle must be initialized and opened. The item key must be initialized based on the key type, key element, and the entity class. If not a positional qualifier, the item key <code>inkey</code> list must be loaded.

crsp_itm_close

Closes databases indicated by a handle, frees all item list structures, and frees the handle itself.

PROTOTYPE:	<code>int crsp_itm_close (CRSP_ITM_HNDL **hndl)</code>
DESCRIPTION:	Frees all item lists and item indexes, clears all calendar and key lists, closes the database, frees the handle set, and frees the handle itself. On completion, the handle will be set to NULL.
ARGUMENTS:	<p><code>CRSP_ITM_HNDL **hndl</code> - Pointer to the item handle to close.</p>

RETURN VALUES:	<p>CRSP_SUCCESS: If the database is successfully closed and all handle data are free</p> <p>CRSP_FAIL: If there is an error in the parameters, inconsistent handle, error closing databases.</p>
SIDE EFFECTS:	<p>If successful, the handle data are emptied:</p> <ul style="list-style-type: none"> • The itm_set database will be closed and the structure cleared. • The itm_grp, itm_idx, and itm_avail arrays will be emptied and all item lists will be freed. • Itm_key and cal_avail arrays will be freed. • The handle itself will be freed and its pointer set to NULL.
PRECONDITIONS	The item handle must be previously opened with <code>crsp_itm_init</code> .

ITEM SELECTION FUNCTIONS

`crsp_itm_load`

Prepare a list from a `full_list` description string.

PROTOTYPE:	<code>int crsp_itm_load(CRSP_ITM_HNDL *hndl, char *full_list, int match_flag)</code>
DESCRIPTION:	Prepare items described by a full list and load them to an item table structure in an item handle. Splits the full list into the global section and the list section and uses <code>crsp_itm_expand_elem</code> on each list element in the list section. This will recursively expand the list elements to fill the structure and apply global qualifiers during the process.
ARGUMENTS:	<p><code>char *full_list</code> - Pointer to a string describing all items to add, used on standard item notation.</p> <p><code>CRSP_ITM_HNDL *hndl</code> - Pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>int match_flag</code> - Flag setting the behavior when an item is found but not the keyset. Values are:</p> <ul style="list-style-type: none"> • <code>CRSP_MATCH_REQUIRED (=0)</code> - if any indicated item and keyset is not found no further items will be added, and <code>CRSP_NOT_FOUND</code> returned. • <code>CRSP_MATCH_FILL (=1)</code> - a dummy item will be created for any item if the item exists but the keyset does not exist for that item in the current database. • <code>CRSP_MATCH_IGNORE (=2)</code> - items will not be added if the keyset is not found, but the return remains <code>CRSP_SUCCESS</code>.
RETURN VALUES:	<p>CRSP_SUCCESS: If successful, and all indicated items loaded according to <code>match_flag</code></p> <p>CRSP_FAIL: Error in parameters, bad list, handle not initialized, or reference data not available.</p>
SIDE EFFECTS:	If successful, the <code>CRSP_ITM_GRP</code> is loaded with all indicated items. A <code>CRSP_ITM</code> is allocated for each item/keyset pair not already loaded. Object pointers are not set by this function.
PRECONDITIONS:	The item handle set must be loaded. The item table must be initialized with an available <code>app_id</code> . The first set in the set list must agree with the <code>app_id</code> .

crsp_itm_load_file

Prepare a list from an item list description file

PROTOTYPE:	<code>int crsp_itm_load_file (CRSP_ITM_HNDL *hndl, char *file_path, char *gbl_list, int match_flag)</code>
DESCRIPTION:	Prepare items described by a listfile and load them to an item table structure in an item handle. Identifies a global section and uses <code>crsp_itm_load_elem</code> on each list element in the file. This will recursively expand the list elements to fill the structure and apply global qualifiers during the process.
ARGUMENTS:	<p><code>char *file_path</code> - pointer to a string containing an input file of data items. Each row in the input file must be a list element.</p> <p><code>char *gbl_list</code> - pointer to a string containing the global information to be applied to all list elements.</p> <p><code>CRSP_ITM_HNDL *hndl</code> - pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>int match_flag</code> - Flag setting the behavior when an item is found but not the keyset. Values are:</p> <ul style="list-style-type: none"> • <code>CRSP_MATCH_REQUIRED (=0)</code> – if any indicated item and keyset is not found no further items will be added, and <code>CRSP_NOT_FOUND</code> returned. • <code>CRSP_MATCH_FILL (=1)</code> – a dummy item will be created for any item if the item exists but the keyset does not exist for that item in the current database. • <code>CRSP_MATCH_IGNORE (=2)</code> – items will not be added if the keyset is not found, but the return remains <code>CRSP_SUCCESS</code>.
RETURN VALUES:	<p><code>CRSP_SUCCESS</code>: If successful</p> <p><code>CRSP_FAIL</code>: If error in parameters, bad list, handle not initialized, or reference data not available.</p>
SIDE EFFECTS:	If successful, the <code>CRSP_ITM_GRP</code> is loaded with all indicated items. A <code>CRSP_ITM</code> is allocated for each item/keyset pair not already loaded. Object pointers are not set by this function.
PRECONDITIONS:	The item handle set must be loaded. The item table must be initialized with an available <code>app_id</code> . The first set in the set list must agree with the <code>app_id</code> . The input file must exist with one list element per row.

crsp_itm_load_printopt

Prepare a list from a groupname print option code.

PROTOTYPE:	<code>int crsp_itm_load_printopt (CRSP_ITM_HNDL *hndl, char *printopt, int match_flag)</code>
DESCRIPTION:	Prepare items described by a print option code describing one group and load them to an item table structure in an item handle.
ARGUMENTS:	<p><code>char *printopt</code> - pointer to a string containing a print option code in the form <code>xx[keyset_string]</code>.</p> <p><code>CRSP_ITM_HNDL *hndl</code> - pointer to the item handle containing the needed set structure information and the current item list.</p> <p><code>int match_flag</code> - flag setting the behavior when an item is found but not in the keyset. Values are:</p> <ul style="list-style-type: none"> • <code>CRSP_MATCH_REQUIRED (=0)</code> – if any indicated item and keyset is not found no further items will be added, and <code>CRSP_NOT_FOUND</code> returned. • <code>CRSP_MATCH_FILL (=1)</code> – a dummy item will be created for any item if the item exists but the keyset does not exist for that item in the current database. • <code>CRSP_MATCH_IGNORE (=2)</code> – items will not be added if the keyset is not found, but the return remains <code>CRSP_SUCCESS</code>.

RETURN VALUES:	CRSP_SUCCESS: if successful. CRSP_FAIL: Error in parameters, opening input file, bad format of global list or input file, or reference data not available.
SIDE EFFECTS:	If successful, the CRSP_ITM_GRP is loaded with all indicated items. A CRSP_ITM is allocated for each item/keyset pair not already loaded. Object pointers are not set by this function.
PRECONDITIONS:	The item handle set must be loaded. The item table must be initialized with an available app_id. The first set in the set list must agree with the app_id. The 2-letter print option code must be known to the app_id. Only groups with grptype of S or D will have non-blank printopt codes available.

crsp_itm_find

Access an individual item that was loaded.

PROTOTYPE:	int crsp_itm_find (CRSP_ITM_HNDL *hndl, char *itm_name, int keyset, CRSP_ITM **foundptr)
DESCRIPTION:	Attach a pointer to a CRSP_ITM that was previously loaded. The CRSP_ITM structure describes the data item and contains the underlying time series, array, or row data.
ARGUMENTS:	CRSP_ITM_HNDL *hndl - Pointer to the item handle containing the needed set structure information and the current item list. char *itm_name - String containing the itm_name to find. int keyset - Keyset to find CRSP_ITM **foundptr - Pointer that will point to the item data found.
RETURN VALUES:	CRSP_SUCCESS: If successful CRSP_NOT_FOUND: If the itm_name and keyset combination are not available CRSP_FAIL: If error in parameters, handle not initialized, or error searching for the item.
SIDE EFFECTS:	If successful, the foundptr will point to a CRSP_ITM with data and information for the desired item and keyset.
PRECONDITIONS:	The item handle set must be initialized, loaded with a list of items, and opened.

crsp_itm_free_list

Resets all item lists previously loaded into a handle.

PROTOTYPE:	int crsp_itm_free_list (CRSP_ITM_HNDL *hndl)
DESCRIPTION:	Resets the handle by freeing all item lists and item indexes.
ARGUMENTS:	CRSP_ITM_HNDL *hndl - pointer to the item handle to reset.
RETURN VALUES:	CRSP_SUCCESS: If successfully frees the data CRSP_FAIL: If error in parameters, inconsistent handle, error emptying the lists.
SIDE EFFECTS:	If successful, the item lists are emptied: itm_list, keyset_list, struct_list. The index arrays are also emptied. New items can be loaded.
PRECONDITIONS:	The item handle must be previously opened with crsp_itm_init.

crsp_itm_is_miss_arrval

Check if a value from a data-object attached to the item is a missing value

PROTOTYPE:	<code>int crsp_itm_is_miss_arrval (CRSP_ITM *itm, int ind*is_miss)</code>
DESCRIPTION:	Checks if the requested element in a data-object attached to the item contains a missing value. <code>is_miss</code> is set to 1 when missing value is detected. Only items of simple (non-structured) types are accepted, while the item's underlying data-object can be of structured data-type, in which case the structure offset is used to extract the item value.
ARGUMENTS:	<p><code>CRSP_ITM *itm</code> - Pointer to the item</p> <p><code>int ind</code> - Index of the data array element to check</p> <p><code>int *is_miss</code> - Pointer to the resulting flag value</p>
RETURN VALUES:	<p><code>CRSP_SUCCESS:</code> If successful, the returned value is initialized and set.</p> <p><code>CRSP_FAIL:</code> If error in parameters, bad item or element index is out-of-range (ignored in case of <code>CRSP_ROW</code>)</p>
SIDE EFFECTS:	
PRECONDITIONS:	The item has to have a valid bound data-object. Structured items are not allowed. Field items of structures are allowed.