WSA API REFERENCE GUIDE

Generated by Moody’s Analytics, SAV

Contents

Chapter 1

Moody's Analytics WSA API Overview

The Moody's Analytics WSA API projects cashflows for CMOs/Remics and Asset-Backed Securities based on scenario information. The scenario information is based on environmental factors (such as market rates), deal-specific characteristics (such as prepayment and default rates) and the status of the deal (which is updated monthly).

1.1 Downloading Library Content

Use the separate download tool and corresponding instructions to download deal files.

1.2 General programming procedure

1.2.1 Part 1 - Initialization of the Library

The first step to projecting cashflows is to initialize the API (automatic in Windows) and let the API know where the data files are located (using set_input_path()) and the locations of any integrated forecast models.

Note

The WSA API is compatible with 32-bit and 64-bit Windows and 32-bit and 64-bit Linux.

1.2.2 Part 2 - Thread Safety

The API has old (single-threaded) as well as new (multi-threaded) levels. To run libraries safely in a multi-threaded environment, user needs to obtain a thread specific identifier for each thread. The API can be used in a thread-safe manner provided:

1. All data and model paths are set in the main thread before creating the deal scenario objects
2. Each deal scenario object has a distinct log file name.
3. The maximum number of deal scenario objects (dso) is set before creating dso's.
4. Each dso is active in only one thread at a time (it can be created in one thread, but processed in another).
5. No functions marked as not thread-safe are used.

Call create_deal_scenario_object(). Thread ID should be passed to every API function working in a given thread. If a thread ID is not created, and NULL is passed as the thread ID, a "default" dso is used. This is not thread safe!

Warning

> Make sure to call release_deal_scenario_object() at the completion of the thread. Failure to do so will result in significant memory leaks.

Concurrent Deals:

Follow the thread-safety guidelines if more than one deal will be open at a time.

1.2.3 Part 3 - Error Handling

Most WSA API functions return an integer value, where a negative integer indicates an error. To obtain the text message of the error, user should call get_deal_error_msg().

Note

> Moody's Analytics strongly encourages users to call this function following every API call.
> To set up a log file, call set_log_options(). If a user wants to control which errors will stop the process or not, call set_error_handling_level().

1.2.4 Part 4 - Loading the Deal

Note

> Loading an SFW deal is the same as loading a CHS deal.

A deal can be loaded by calling open_deal_ex(). User provides input fields by calling open_deal_ex().

Important inputs:

- dealid - Name of the deal/CHS or the SFW file provided by Moody's Analytics
- actual_coll - Controls level of collateral; value 0 indicates repline-level, value 1 indicates loan-level
- settlement_date - Has to be provided in format MM/DD/YY
- get_markit_id - Helps user convert the CUSIP ISIN to Moody's Analytics deal name and tranche name.
- get_deal_info - Allows the user to examine the status of the deal loading (particularly whether the deal was loaded under the loan level, repline level, the date of the last update file used, etc.).

Warning

> Make sure to call close_deal_ex() after you have finished working with the open deal. Failure to do so will generate significant memory leaks.

1.2.5 Part 5 - Reviewing the Deal Collateral and Capital Structures

1.2.5.1 Viewing Bond Information

Call view_all_bonds_ex() to see all bonds in a deal.

1.2.5.2 Viewing Group Information

Call view_coll_groups() to see all collateral groups in the deal.

1.2.5.3 Viewing Bond Collateral

Call view_colls_ex() to load all collateral data into array.

Moody's Analytics encourages users to use these functions:

- obtain_collat_iterator()
- get_next_collat()

1.2.5.4 Viewing Reremic Deals

Call view_reremic_deals() to view reremic deals.

1.2.5.5 Viewing Parameters of Reremic Deals

Most of the functions in the API have a counterpart for manipulating the reremic deals. For example, view_reremic_-colls_ex() allows users to see all of their collaterals of the reremic deal designated with deal ID. Some new functions can work on both parent and children deals. For example, obtain_collat_iterator() has the following parameter: reremic_deal_id_or_null. This indicates whether to access a particular child deal or a parent deal.

1.2.6 Part 6 - Running the Cashflows

1.2.6.1 Levels of Control

By default, Moody's Analytics WSA API uses its own collateral files and user supplied assumption parameters to amortize the collateral and distribute cashflows between bonds. User can pre-empt some of those functions. For example, user can provide their own collateral information using the function: replace_collateral(). User also can decide to generate collateral cashflows by themselves and rely on the WSA API only for distributing cashflow among the bonds. The functions are:

- set_custom_amortization_ex()
- set_collateral_flow_ex()

1.2.6.2 Setting Collateral Assumptions

User can set deal level or collateral level assumptions using the following functions:

- set_prepayments_ex()
- set_defaults_ex()
- set_recoveries_ex()
- set_recovery_lag_ex()
- set_service_advances_ex()

User can also use pr collateral per period callback functions to set the assumptions. Use the following set of functions:

- install_per_period_assump_cb()
- install_collat_assump_cb()
- set_user_data_for_cb()
- get_user_data_for_cb()
- install_user_cleanup_cb()

1.2.6.3 Setting Interest Rates

Call set_rate_ex() to set a particular forward rate.

Call get_rates() to see which rates are required by a given deal.

Note

If a necessary rate is not supplied, value of zero is used. No error message is generated.

1.2.6.4 Running the Cashflows

After all of the assumptions are set, user should run this function run_deal_ex() to request the API to perform the calculation.

1.2.7 Part 7 - Force Obtaining Results of the Cashflows

1.2.7.1 Obtaining Collateral Cashflow

Call get_cf_date() use this function to obtain dates of the collateral cashflow.

Call get_collateral_flow_ex() to obtain values of the collateral cashflow.

Call get_collateral_flow_ex1() to obtain all collateral cashflow information in one function.

1.2.7.2 Obtaining Bond Cashflow

Call get_cf_date() to obtain dates of the bond cashflow.

Call get_bond_flow_ex() to obtain values of the bond cashflow.

Call get_bond_flow_ex1() to obtain all bond cashflow information in one function. The function also adds historical cashflows to which some zero delay tranches are entitled.

The following chapters describes the files and the preferred locations for installing them, followed by an explanation of the pre-processing directives, structures used, an overview of the functions, and detailed information about each function.

Chapter 2

WSA API Files

The WSA API consists of a dynamic library containing the subroutines and data files that are accessed by those subroutines. The structures, constants, and function prototypes are defined in the C/C++ header files. The files are installed by extracting them to the appropriate folders/directories. The files do not need to be registered.

The library can be either linked to at compile time, or loaded via code during the run. An import library is provided for Windows users who link at compile time.

On Windows systems these are usually put in the System32 sub-folder. The dynamic library should be in a path automatically checked by the system unless you are loading from your code and fully qualify the path to the library.

1. Header files

File Name	Description
wsa.h	The header file for the WSA API. This file should be included in the client project.
ccmo.h	The header file from the legacy CHS API. It is included through wsa.h.
indextypes.h	The header file contains the enumeration of index rate types as well as some collateral enumerators.

2. Core Library files

File(Windows)	File(Linux)	Description
cmo_W32.dll	libCmoLinux.so	This is the underlying CHS API library. It should be placed in the directory containing the client application executable file.
wsa.dll	libwsa.so	This is the WSA API library. It should be placed in the directory containing the client application executable file.
sfw.dll	libsfw.so	This is the underlying SFW API library. It should be placed in the directory containing wsa.dll.
cdonet.dll	libcdonet.so	This is the underlying CDOnet API library. It should be placed in the directory containing wsa.dll.
wsa_nw.dll	N/A	This is a convenient .Net wrapper which encapsulates the interop calls to the underlying wsa.dll. If you are going to call the API from a .Net application, this dll should be referenced in the project.

wsa.lib	N/A	This is the WSA API library file (for Windows only). This library should be referenced in the client project and the project should be recompiled to be able to call the WSA API. ANY EXISTING REFERENCES TO cmo_W32.lib SHOULD BE REMOVED FROM THE CLIENT PROJECT.

3. ADCo Module file

File(Windows)	File(Linux)	Description
WSAAdcoProvider.dll	libWSAAdcoProvider.so	ADCo integration module library.

4. AFT Module file

File(Windows)	File(Linux)	Description
WSAAftProvider.dll	N/A	AFT integration module library.

5. PA (Portfolio Analyzer-ABS) Module files

File Name	Description
Abs1ApiLibrary.dll	PA(Portfolio Analyzer-ABS) API library (from PA providers).
SupportData.db	Economic database used by PA calculation (from PA providers). It should be placed in the directory that is set via call to set_pa_data_path() API method.
sqlite3.dll	Sqlite3 driver.

6. MPA (Mortgage Portfolio Analyzer) Module files

File Name	Description
Mpa2ApiLibrary.dll	MPA(Mortgage Portfolio Analyzer) API library (from MPA providers).
SupportData.db	Economic database used by MPA calculation (from MPA providers). It should be placed in the directory that is set via call to set_mpa_data_path() API method.
sqlite3.dll	Sqlite3 driver.

7. Deal file

File Name	Description
∗.SFW	All SFW/CDO deal files need to be placed in the directory set via the API set_input_path() function.
∗.CHS	All CHS deal files need to be placed in the directory set via the API set_input_path() function.

8. Misc File

File Name	Description
WSA_API.DB	This database file (downloaded via download tool) contains deal and CUSIP mapping information. It should be placed in the directory that is set via call to set_input_path() API method.

Chapter 3

WSA API Core Functions

The following is a list of functions by category. Each function is described in detail, along with an example of its use, in the Appendix.

Note

> The balances of the functions are used to obtain descriptive information, to set scenarios before a run, or to obtain cashflows after a run: create_deal_scenario_object(), open_deal_ex(), run_deal_ex(), close_deal_ex(), and release_deal_scenario_object().

All functions can be called from either Windows or Unix unless noted with one of the following:

- Windows Only Function
- Unix/Linux Only Function

Deal Scenario Objects (dso's) are used to allow more than one deal to be open at a time and to run multi-threaded analysis. Each dso can contain one deal at a time, and each dso must be processed in only one thread at a time.

If the same deal is to be run with different scenarios in different threads there must be a separate dso for each thread.

Note

> If a dso is not created, and NULL is passed as the tid, a "default" dso is used. This is not thread-safe!

Warning

> Not all functions are supported on both of the deal libraries. If you install updated versions of the WSA API as they're released then you do not have to programmatically check if the function supports the deal you're opening. For each new release of the WSA API we will enable additional supported SFW deals.

Note

> For your reference we've indicated which functions are supported on which libraries.

3.1 Basic Functions

Function	Description	Library

close_deal_ex()	Close the deal and release resources. The deal/scenario object is still available for processing another deal.	ALL
create_deal_scenario_object()	Create a deal/scenario object	ALL
get_deal_error_msg()	Obtains text of the error generated by the previous API function. If no error 0 is returned	ALL
get_sdk_build_version()	This function returns the Software release version of this WSA API build .	ALL
install_collat_assump_cb()	Installs collateral assumption call back function.	CDOnet,CHS,SFW
install_per_period_assump_cb()	Installs per period assumption call back function.	CDOnet,CHS,SFW
install_user_cleanup_cb()	installs user clean up function	CDOnet,CHS,SFW
open_deal_ex()	Open the specified deal and return descriptive information	ALL
open_pool_from_file()	This function opens a single pool. This function only applies to CHS deals. It will return SUCCESS if an SFW deal is called.	ALL
run_deal_ex()	Project cashflows for a previously opened deal based on scenario information set by the user.	ALL
release_deal_scenario_object()	Release the dso and free the resources.	ALL
set_input_path()	Tells the WSA API where the chs and sfw data files are located. The total length of the string cannot exceed 60 characters.	ALL
set_log_options()	Sets the options for logging messages (file, pop-up and etc.). Pop-up is for Windows only.	ALL
write_log()	This function writes the given message to the log file.	ALL

3.2 Obtain Descriptive Information

Function	Description	Library
get_markit_id()	Retrieves the deal ID and bond ID for either a cusip or isin.	ALL
get_markit_id1()	Returns the deal and bond id for an industry-standard bond identifier. Comparing with get_markit_id(), this function reports error message through its last 2 parameters.	ALL

get_moodys_id()	Returns the deal and bond id for an industry-standard bond identifier. This function retrieves dealid and bondid of arbitrary length. (For SFW deals, bondid can exceed 7 characters)	ALL
get_deal_account_avail()	This method retrieves the name of the accounts in the SFW deal.	CDOnet, SFW
get_deal_info()	Populates the user allocated buffer with surveillance data for a specific month, if the surveillance data is available.	CDOnet,CHS,SFW
get_deal_update_id()	Returns SFW deal library update ID for SFW deals.	CDOnet,CHS,SFW

Bond Information

Function	Description	Library
get_deal_surv_data()	This function retrieves the deal surveillance data as of the month and year provided in the format YYYYMM.	CHS
get_bond_band()	Get the pac band info for the bond (pricing WAL, low speed, high speed)	CDOnet,CHS,SFW
get_bond_by_index_ex()	Provides descriptive information about one bond (specified by numeric index).	CDOnet,CHS,SFW
get_bond_by_name_ex()	Provides descriptive information about one bond (specified by bond name).	CDOnet,CHS,SFW
get_bond_day_cal_cur_ex()	Provides the day count, calendar type, and currency for the specified bond.	CDOnet,CHS,SFW
get_bond_index_ex()	Provides the numeric index for the specified bond name.	CDOnet,CHS,SFW
get_bond_info_by_index()	Get bond information by its index.	CDOnet,CHS,SFW
get_bond_info_by_tranche()	Get the bond info from its tranche name.	CDOnet,CHS,SFW
get_bond_misc_ex()	Provides additional information that is not in the CCMO_BONDS_S structure for the requested bond.	CDOnet,CHS,SFW
view_all_bonds_ex()	Provides descriptive information about all bonds in a deal.	CDOnet,CHS,SFW

Collateral Information

Function	Description	Library
get_pool_by_index_ex()	Provides descriptive collateral information about one piece of collateral (specified by numeric index).	CDOnet,CHS,SFW

get_pool_ptr_by_index_ex()	Provides a pointer to a CCMO_POOL_INFO structure for one piece of collateral (specified by numeric index).	CDOnet,CHS,SFW
view_colls_ex()	Provides more information on all collateral backing a deal.	CDOnet,CHS,SFW
get_average_collat()	Obtains average collateral for a deal	CDOnet,CHS,SFW
get_average_collat_for_managed-_code()	This function is for managed code. All structures should have been allocated by the caller. There is no memory allocated by SDK.	CDOnet,CHS,SFW
get_group_info()	Obtains collateral group info	CDOnet,CHS,SFW
get_moodys_pool_group_id()	Retrieves Moody's poolgroup ID.	SFW

Collateral iteration and replacement

Function	Description	Library
install_pool_cashflow_cb()	Installs the user defined pool cashflow callback function.	CDOnet,CHS,SFW
get_next_collat()	Using iterator get next collateral	CDOnet,CHS,SFW
get_next_collat_for_managed_-code()	This function is for managed code, all structures should have been allocated by the caller. There is no memory allocated by SDK. Function will return 0 if collateral is successfully loaded, 1 if end of collateral list, and negative values if errors.	CDOnet,CHS,SFW
obtain_collat_iterator()	Start iteration of the collateral	CDOnet,CHS,SFW
view_coll_groups()	Reports all available collateral groups	CDOnet,CHS,SFW
replace_collateral()	Replaces collateral of the deal with one provided by the user	CDOnet,CHS,SFW

Miscellaneous(part 1)

Function	Description	Library
dayt_to_str()	Converts a date to a string (mm/dd/yy).	ALL
deal_has_underlying_deal()	Check if the deal has underlying deals.	CDOnet,CHS,SFW
get_cleanup_call_ex()	Provides information on the type of call for the deal.	CDOnet,CHS,SFW
get_dates_from_upd_ex()	Returns an array of available deal status dates for the specified deal (yyyymm).	CDOnet,CHS,SFW
get_deal_calc_level()	Gets the deal calculation level.	CDOnet,CHS,SFW
get_deal_issuer_type()	Provides the deal issuer and type.	CDOnet,CHS,SFW

Miscellaneous(part 2)

Function	Description	Library
get_hist_data_ex()	Returns the historical factors and coupons for a bond.	CDOnet,CHS,SFW
get_input_path()	Returns the current path to the Moody's Analytics deals.	ALL
get_longest_ex()	Returns the maximum vector length required for setting scenarios. This will also be the last period in which there is any cashflow.	CDOnet,CHS,SFW
get_surv_avail_YYYYMMs()	This function retrieves the deal surveillance data as of the month and year provided in the format YYYYMM.	CHS
get_surveillance_data()	Populates the user allocated buffer with surveillance data for a specific month, if the surveillance data is available.	CHS
get_triggers_avail()	Provides information describing the triggers in the deal.	CDOnet,CHS,SFW
get_trigger_status()	Provides information about the requested trigger's status, See get_triggers_avail() for an explanation of triggers.	CDOnet,CHS,SFW
get_user_data_for_cb()	Retrieves the band information for the bond in the specified underlying deal (pricing WAL, low speed and high speed).	CDOnet,CHS,SFW

3.3 Set Library

Function	Description	Library
get_current_deal_engine()	After a deal has been opened, call this function to confirm which library it's from.	ALL
set_engine_preference()	Specify which library to use in the event that a deal exists in both.	ALL

3.4 Set Scenarios

Set Interest Rates

Function	Description	Library
get_rate_ex()	Get the rate used for the given index.	CDOnet,CHS,SFW
get_rates_ex()	Returns information on which rates are required for a deal (includes bonds and collateral).	CDOnet,CHS,SFW
get_required_rate_codes()	This function can be used to retrieve the codes of required index rates.	CDOnet,CHS,SFW

set_missing_interest_rates_-handling()	This function sets the rule to handle the case where some interest rates are missing.	ALL
set_rate_ex()	Sets the market rate for a specified market index. The rate may be either a constant or a vector.	CDOnet,CHS,SFW

Set Prepayments

Function	Description	Library
set_prepayments_ex()	Sets the prepayment curve for either all collateral or an individual piece of collateral, with the ability to reset the underlying deals if a reremic.	CDOnet,CHS,SFW

Set Credit Sensitivity

Function	Description	Library
is_credit_sensitive_ex()	Used to determine if a deal or piece of collateral is credit sensitive.	CDOnet,CHS,SFW
set_deal_account_default()	Turns the account on and off by setting the account default status.	CDOnet,CHS,SFW
set_addit_group_delinquencies()	Set the group level delinquency for a specific type.	CDOnet,CHS,SFW
set_default_from_ex()	Determines whether defaults are from the current balance, zero-prepay balance, or the balance as of the deal status date ("original") with the option to apply to any underlying deals. Current balance is used if not set.	CHS
set_defaults_ex()	Sets the default curve for either all collateral or an individual piece of collateral, with the ability to reset the underlying deals if a reremic.	CDOnet,CHS,SFW
set_recoveries_ex()	Sets the recovery curve for either all collateral or an individual piece of collateral, with the ability to reset the underlying deals if a reremic. Recoveries are the inverse of losses (Recovery % + Loss % = 1.0).	CDOnet,CHS,SFW

set_recovery_lag_ex()	Sets the recovery lag for either all collateral or an individual piece of collateral, with the ability to reset the underlying deals if a reremic.	CDOnet,CHS,SFW
set_service_advances_ex()	Sets the type of servicer advances (none, interest only, both), with the option to apply to underlying deals (if they exist).	CDOnet,CHS,SFW

Miscellaneous

Function	Description	Library
clean_up_call_ex()	If set to true, the deal will run to call. Otherwise the deal will not be called. The value can be applied to all underlying deals if a reremic.	CDOnet,CHS,SFW
set_deferment_rates()	Sets deferment rates for SFW SLABS deals	SFW
set_forbearance_rates()	Sets forbearance rates for SFW SLABS deals	SFW
set_moodys_credit_model_-settings()	Loads Moody's DPLC Run At mode assumptions for SFW deals	CDOnet,CHS,SFW
set_trigger_override()	Overrides the trigger calculations in the deal for each period. See get_triggers_avail() for an explanation of triggers.	CDOnet,CHS,SFW

3.5 Set Custom Amortization

Function	Description	Library
set_collateral_flow_ex()	Sets the specified collateral flow values for the deal or specified collateral group.	CDOnet,CHS,SFW
set_custom_amortization_ex()	Sets the type of custom amortization (by deal or by collateral group).	CDOnet,CHS,SFW

3.6 Set Miscellaneous

Function	Description	Library
set_deal_search_mode()	Set the mode for deal search: either DEAL_SEARCH_FROM_FILE or DEAL_SEARCH_FROM_MEMO-RY.	ALL
set_error_handling_level()	Set the error handling level for processing.	ALL

3.7 Obtain Cashflows

Function	Description	Library
calc_cashflow_offset_ex()	Cashflows are projected as of the deal status date. This calculates the offset into the cashflows required for a given transaction settlement date and the number of days of accrued interest.	CDOnet,CHS,SFW
get_bond_flow_ex()	Provides a pointer to the requested cashflow for the specified bond.	CDOnet,CHS,SFW
get_bond_flow_ex1()	Provides a pointer to the MARKIT_BOND_CASHFLOW structure which holds all bond cashflow information	CDOnet,CHS,SFW
get_bond_flow_ex1_for_managed-_code()	A variation of function get_bond_flow_ex1(), and the difference is the structures used in this function are all using static memory.	CDOnet,CHS,SFW
get_cf_date()	Returns the payment date corresponding to the requested period.	CDOnet,CHS,SFW
get_collateral_flow_ex()	Provides a pointer to the requested cashflow for deal or the specified collateral group.	CDOnet,CHS,SFW
get_collateral_flow_ex1()	Populates the user allocated structure of type MARKIT_COLLAT_CASHFLOW with collateral cash flow data.	CDOnet,CHS,SFW
price_bond()	Calculates cashflow analytics for a given bond	CDOnet,CHS,SFW
set_bond_cf_mode()	Set the bond cash flow mode and payment dates type.	CDOnet,CHS,SFW
set_deal_calc_level()	Set the deal calculation level: CALC_LEVEL_BASIC or CALC_LEVEL_FULL	CDOnet,CHS,SFW

Chapter 4

WSA API Reremic Functions for Underlying Deals

The format is similar, with reremic_ after the get_ , set_ or view_ (view_reremic_colls_ex(), rermic_is_credit_-sensitive).

4.1 Obtain Descriptive Information

Function	Description	Library
view_reremic_deals()	Provides descriptive information about all underlying deals for the opened deal.	CDOnet,CHS,SFW

Bond Information

Function	Description	Library
get_reremic_bond_band()	Get the pac band info for the specified bond in the underlying deal (pricing WAL, low speed, high speed)	CHS,SFW
get_reremic_bond_misc()	Provides additional information that is not in the CCMO_BONDS_S structure for the requested bond in the underlying deal.	CDOnet,CHS,SFW

Collateral Information

Function	Description	Library
get_reremic_pool_ptr_by_index()	Provides descriptive collateral information about one piece of collateral (specified by numeric index) in the underlying deal.	CHS,SFW

view_reremic_colls_ex()	View extended collateral information for the collateral in an underlying deal.	CHS,SFW

Miscellaneous

Function	Description	Library
get_reremic_triggers_avail()	Provides information describing the triggers in the underlying deal.	CHS,SFW

4.2 Set Scenarios

Set Prepayment

Function	Description	Library
set_reremic_prepayments()	Sets the prepayment curve for either all collateral or an individual piece of collateral for an underlying deal.	CHS,SFW

Set Credit Sensitivity

Function	Description	Library
set_reremic_default_from()	Determines whether defaults in the underlying deal are from the current balance, zero-prepay balance, or the balance as of the deal status date (“original”). Current balance is used if not set.	CHS
set_reremic_defaults()	Sets the default curve for either all collateral or an individual piece of collateral for an underlying deal.	CHS,SFW
set_reremic_recoveries()	Sets the recovery curve for either all collateral or an individual piece of collateral for an underlying deal. Recoveries are the inverse of losses (Recovery % + Loss % = 1.0).	CHS,SFW
set_reremic_recovery_lag()	Sets the recovery lag for either all collateral or an individual piece of collateral for an underlying deal.	CHS,SFW
set_reremic_service_advances()	Sets the type of servicer advances (none, interest only, both) for the underlying deal.	CHS,SFW

Miscellaneous

Function	Description	Library
set_reremic_trigger_override()	Overrides the trigger calculations in the underlying deal for each period.	CHS,SFW

4.3 Set Scenarios using user call back mechanism

Call back signatures

Name	Description
USER_CLEANUP_CB	User function which can be called by the API to let user clean up user data registered through call to set_user_data_for_cb()
COLLAT_ASSUMP_CB	User function called by the API to get assumptions for every collateral
PER_PERIOD_ASSUMP_CB	User function called by API for each simulation period of each collateral to provide user with current simulation state and get from user assumptions for next period

4.4 Registering call backs functions

Registering call backs functions

Function	Description	Library
install_per_period_assump_cb()	Registers PER_PERIOD_ASSUMP_CB type of functions with the API	CDOnet,CHS,SFW
install_collat_assump_cb()	Registers COLLAT_ASSUMP_CB type of function with the API	CDOnet,CHS,SFW
install_user_cleanup_cb()	Registers USER_CLEANUP_CB type of function with the API	CDOnet,CHS,SFW
set_user_data_for_cb()	Allows the user to register some data with the API which will be passed to all call back functions. Allows the user to maintain a state.	CHS,SFW

4.5 Obtain Reremic Cashflows

Obtain Reremic Cashflows

Function	Description	Library
get_reremic_trigger_status()	Provides information about the requested trigger status in the underlying deal.	CHS,SFW

Chapter 5

AD&Co© Reference

The WSA API is compatible with AD&Co©'s credit model version 1.9. See this section for functions and Chapter 2 for a list of the related files used in the WSA API.

5.1 Structures

Structures	Description
MarkitAdcoPrepayModelDials	Supplements structure MarkitAdcoTuningParam, and is optional.
MarkitAdcoDefaultModelDials	Supplements structure MarkitAdcoTuningParam, and is optional.
MarkitAdcoTuningParam	The main tuning parameters for the ADCO default model.

5.2 Functions

Function	Description
SetupADCOModel()	Install ADCo model to WSA API.
RemoveADCOModel()	Uninstall ADCo model from WSA API.
ResetADCOScenario()	Reset all interest rates and set the parameters in ADCo model.
ResetADCOInterestRates()	Reset all interest rates in ADCo model.
ResetADCOHpiRates()	Reset HPI rates in ADCo model.
GetADCOVersion()	Get The ADCO version.
SetTuningParam()	Set tuning parameters. This function should be called from LOAN_TUNNING_CB.
GetCurrentLoanType()	Get the type of loan that are dealed with. This function should be called from LOAN_TUNNING_CB.

Chapter 6

MPA Credit Model

6.1 MPA Overview

Mortgage Portfolio Analyzer (MPA) is an analytic platform for assessing the credit risk of wholeloan residential mortgage portfolios and collateral pools underlying residential mortgage-backed security (RMBS) structures.

MPA contains a macro-economic simulator and loan-level models for estimating probabilities of default, prepayment and loss-given-default (severity).

6.1.1 Prerequisite

- SupportData.db

  When running MPA analysis, API require a sqlite3 database file "SupportData.db" to be existing in path specify by set_mpa_data_path().

  If user don't specify the database, API will automatically search in deal folder (which set by set_input_path()), i.e , if set_input_path("/tmp"), then, API will look SupportData.db in '/tmp/MPA/data/SupportData.db'.

- Mpa2ApiLibrary.dll

  This file should be in same folder with wsa.dll.

- Mid course adjustment data (if API need to calls set_mpa_mid_course_adj()).

- WSA API version $>$ 2.0 (MPA simulation require WSA API $>$ 2.1)

6.1.2 Steps for MPA Analysis

Step 1: set credit model to MPA

Calling set_moodys_credit_model_settings() with MOODYS_MPA_SETTINGS to notify API to run with MPA credit model.

```
*  ...
*    open_deal_ex(tid, pCmo);
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*  ...
*
```

Note

MPA credit model only applies to RMBS deal.

Step 2: set MPA analysis type

There are 4 ways to run MPA analysis: (set by calling set_mpa_analysis_type())

- Pre-define Scenario

  Using macro eoconmics forecase data from SupportData.db

- Custom Scenario

  API provide interfaces that allow user to run MPA with their own economics scenarios.

- Loss Simulation

  Perform loss simulation . Result cashflow is averaged of all paths.

- Custom Scenario Simulation

  Simulate N path of economy scenarios, and calculate cashflows base on different economy path. Result cashflow is averaged of all paths.

Step 3: set MPA analysis assumptions & settings

Set MPA analysis type ,credit assumptions and simulation length before MPA run.

Step 4: Run and get cashflow result

After call run_deal_ex(), API will start to run MPA simulation.

6.2 Set MPA Assumptions

6.2.1 Mid Course Data

Mid Course Data(by set_mpa_mid_course_adj(), to enable MPA to run with mid-course adjustment data.

Users can indicate whether they want the selected analysis to take into consideration the actual prepayment, default and severity experience of the pool.

This option is most useful for evaluating portfolios of seasoned loans for which historical performance information is available.

By default, API won't use mid-course data adjustment.

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    set_current_mpa_scenario(tid, 3);
*
*    // enable API use mid course adjustment data
*    set_mpa_mid_course_adj(tid, True);
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.2.2 Multiplier

Default Multiplier (by set_mpa_multiplier(), which will set multiplier for prepayment , default and severity vector.)

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    set_current_mpa_scenario(tid, 3);
*
*    double default_factor = 0.9;
*    // scale down 10% of defaults on all loans
*    set_mpa_multiplier(tid, MPA_MULTIPLIER_DEFAULT, 0, &
      default_factor, -1);
*
*    double prepay_factor = 1.1;
*    // scale up 10% of prepayments on all loans
*    set_mpa_multiplier(tid, MPA_MULTIPLIER_PREPAY, 0, &
      prepay_factor, -1);
*
*    double severity_factor = 1.15;
*    // scale up 15% of severity on all loans
*    set_mpa_multiplier(tid, MPA_MULTIPLIER_SEVERITY, 0, &
      severity_factor, -1);
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.2.3 Recovery Lag

Recovery Lag (by set_mpa_recovery_lag_by_state() which will set recovery lag base on judicial or non-judicial state,by set_mpa_recovery_lag() which will set recovery lag on deal or loan level)

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    set_current_mpa_scenario(tid, 3);
*
*    //set recovery lag in judicial state with 10, and recovery in non-judicial state with 15
*    set_mpa_recovery_lag_by_state(tid, 10, 15 );
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.2.4 Haircut

Haircut (by set_mpa_haircut(), this function will set the haircut value in form of vector or constant.)

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    set_current_mpa_scenario(tid, 3);
*
*    //set haircut value with 0.05 over all periods.
*    double haircut_value = 0.05 ;
*    set_mpa_haircut(tid, 0, &haircut_value, 0);
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.2.5 Simulation Length

Simulation Length(by set_mpa_simulation_length()), this function will set length of projection periods.)

```
*  ...
*    open_deal_ex(tid, pCmo);
*
```

```
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    set_current_mpa_scenario(tid, 3);
*
*    //set MPA simulation length with 120 periods.
*    set_mpa_simulation_length(tid, 120);
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.2.6 Default Loan Data

Default Data (by set_mpa_default_loan_data(), this function will set default value for a missing mortgage attribute when there is a missing value for a loan.)

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    set_current_mpa_scenario(tid, 3);
*
*    //set defalut LTV value for loans which don't have value in this field
*    set_mpa_default_loan_data(tid, "LTV","75");
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.3 MPA Single Path Analysis

6.3.1 Single Path Setting

To use single path analysis, user should supply set_mpa_analysis_type() with MPA_MEDC_SINGLE_PATH.

```
*  ...
*    open_deal_ex(tid, pCmo);
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    // call set_mpa_analysis_type() with MPA_MEDC_SINGLE_PATH
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    set_current_mpa_scenario(tid, 3)
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.3.2 Single Path: Pre-define Scenario

SupportData.db file contains serveral sets of macro forecasted economic indicators. User can choose(by calling set_current_mpa_scenario()) one of the scenario and run MPA against with.

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    // call set_mpa_analysis_type() with MPA_MEDC_SINGLE_PATH
*    set_mpa_analysis_type(tid, MPA_MEDC_SINGLE_PATH);
*    // set Scenario 3 for MPA single path run.
*    set_current_mpa_scenario(tid, 3)
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.3.3 Single Path: Custom Scenario

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
     MOODYS_MPA_SETTINGS,1);
*
*    // set MPA_CUST_MEDC_SINGLE_PATH for MPA Custom Scenario Simulation
*    set_mpa_analysis_type(tid, MPA_CUST_MEDC_SINGLE_PATH);
*
*    // initialize user define Unemployment rate
*    int UsUemployment_year[] = {2013,2014,2014,2014,2014,2015,2015,2015,2015,2016,2016,2016,2016,2017,2017
     ,2017,2017,2018,2018,2018};
*    int UsUemployment_quarter[] = {4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3};
*    double UsUemployment_value[] = {7.89,7.85,7.77,7.56,7.67,7.42,7.39,7.31,7.26,7.29,7.33,7.28,7.25,7.17,
     7.21,7.19,7.14,7.03,7.01,6.88};
*
*    // set user defined Unemployment rate
*    set_mpa_custom_scenario(tid, "UNEMPLOYMENT","US", UsUemployment_year,
     UsUemployment_quarter, UsUemployment_value,20);
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.3.4 Getting Simulated Cashflow

In single path mode, both cashflows from collaterals and bonds can be retrieved as normal deal run. (see How to get cashflows result)

6.4 MPA Simulation Analysis

There are 2 types of simulation in MPA module

- Loss Simulation (call set_mpa_analysis_type() with MPA_LOSS_SIMULATION)
- Custom Scenario Simulation (call set_mpa_analysis_type() with MPA_CUST_MEDC_SIMULATION)

6.4.1 Simulation Setting

Output Simulation Cashflows Files

set_mpa_simulation_output_path() is used to set path for API to output simulation cashflows.

Warning

output cashflow files could be very large, please use this function cautiously.

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
     MOODYS_MPA_SETTINGS,1);
*    set_mpa_analysis_type(tid, MPA_LOSS_SIMULATION);
*    //Cashflow file will be output to C:\tmp
*    set_mpa_simulation_output_path("C:\tmp");
*    set_current_mpa_scenario(tid, 3);
*    run_deal_ex(tid, pCmo);
*  ...
*
```

Set Number of Simulation Paths

By default, API will run 10000 paths for a single simulation run. API user can specify number of path base on their own requirement.

```
*  ...
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    // set 1000 paths in the simulation
*    set_mpa_simulation_path_num(tid, 1000);
*  ...
*
```

6.4.2 Simulation Type: Loss Simulation

User can call set_mpa_analysis_type() with MPA_LOSS_SIMULATION to instruct API to run loss simulation.

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*    // set MPA_LOSS_SIMULATION for MPA Loss Simulation
*    set_mpa_analysis_type(tid, MPA_LOSS_SIMULATION);
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.4.3 Simulation Type: Custom Scenario Simulation

To use custom scenario, user should:

1. call set_mpa_analysis_type() with MPA_CUST_MEDC_SIMULATION
2. set custom macro economic data

Note

Setting Custom scenario simulation is almost same with setting single path custom scenario except that calling set_mpa_analysis_type() with MPA_CUST_MEDC_SIMULATION.

```
*  ...
*    open_deal_ex(tid, pCmo);
*
*    set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS,1);
*
*    // set MPA_CUST_MEDC_SIMULATION for MPA Custom Scenario Simulation
*    set_mpa_analysis_type(tid, MPA_CUST_MEDC_SIMULATION);
*
*    // initialize user define Unemployment rate
*    int UsUemployment_year[] = {2013,2014,2014,2014,2014,2015,2015,2015,2015,2016,2016,2016,2016,2017,2017
      ,2017,2017,2018,2018,2018};
*    int UsUemployment_quarter[] = {4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3};
*    double UsUemployment_value[] = {7.89,7.85,7.77,7.56,7.67,7.42,7.39,7.31,7.26,7.29,7.33,7.28,7.25,7.17,
      7.21,7.19,7.14,7.03,7.01,6.88};
*
*    // set user defined Unemployment rate
*    set_mpa_custom_scenario(tid, "UNEMPLOYMENT","US", UsUemployment_year,
      UsUemployment_quarter, UsUemployment_value,20);
*
*    run_deal_ex(tid, pCmo);
*  ...
*
```

6.4.4 Getting Simulated Cashflow

In simulation mode(loss simulation or custom scenario simulation), both cashflows from collaterals and bonds are aggregated in average.

API user use ordinary way to retrive result cashflow from simulation.(see How to get cashflows result)

Chapter 7

PA Credit Model

7.1 PA Overview

7.1.1 Prerequisite

- SupportData.db

  When running PA analysis, API require a sqlite3 database file "SupportData.db" to be existing in path specify by set_pa_data_path().

  If user don't specify the database, API will automatically search in deal folder (which set by set_input_path()), i.e , if set_input_path("/tmp"), then, API will look SupportData.db in '/tmp/PA/data/SupportData.db'.

- Abs1ApiLibrary.dll

  This file should be in same folder with wsa.dll.

- WSA API version $>$ 2.0

7.1.2 Steps for PA Analysis

Step 1: set credit model to PA

Calling set_moodys_credit_model_settings() with MOODYS_PA_SETTINGS to notify API to run with MPA credit model.

```
*  ...
*    open_deal_ex(tid, pCmo);
*    set_moodys_credit_model_settings(tid,
      MOODYS_PA_SETTINGS ,1);
*  ...
*
```

Note

PA credit model only applys to ABS deals.

Step 2: set PA analysis type

There are 2 ways to run PA analysis: (set by calling set_pa_analysis_type())

- Pre-define Scenario

  Using macro eoconmics forecase data from SupportData.db

- Custom Scenario

  API provide interfaces that allow user to run PA with their own economics scenarios.

Step 3: set PA analysis scenarios and assumptions

- Set pre-defined scenario by calling set_current_pa_scenario()
- Set custom scenario data by calling set_pa_custom_scenario()
- Set Pool default loan value by calling set_pa_default_pool_data()

Step 4: Run and get cashflow result

After call run_deal_ex(), API will start to run deal with PA scenarios(either custom or pre-defined scenarios).

7.2 Set PA Pool Default Loan Values

By calling set_pa_default_pool_data(), user are able to set default loan attributes for loans that have missing values. For all avaialbe loan fields, please refer to set_pa_default_pool_data() in reference guide.

Note

Type of value to be set is "const char *"

```
*   ...
*     open_deal_ex(tid, pCmo);
*
*     set_moodys_credit_model_settings(tid,
       MOODYS_PA_SETTINGS, 1);
*     set_pa_analysis_type(tid, PA_MEDC_SINGLE_PATH);
*     // set loan field "WAFICO" with default value "650"
*     set_pa_default_pool_data(tid, "WAFICO", "650");
*
*     run_deal_ex(tid, pCmo);
*     // get cashflow result code here
*   ...
*
```

7.3 PA Single Path Analysis

7.3.1 Single Path: Pre-Define Scenario

User can run PA with a pre-defined scenario.

```
*   ...
*     open_deal_ex(tid, pCmo);
*     set_moodys_credit_model_settings(tid,
       MOODYS_PA_SETTINGS, false);
*     set_pa_analysis_type(tid, PA_MEDC_SINGLE_PATH);
*
*     //get total available pre-defined scenarios
*     int num_of_scenarios = get_pa_scenarios(tid, NULL);
*     // set 3rd of pre-defined scenario to run PA
*     set_current_pa_scenario(tid, 3);
*
*     run_deal_ex(tid, pCmo);
*     // get cashflow result code here
*     close_deal_ex(tid, pCmo);
*   ...
*
```

7.3.2 Single Path: Custom Scenario

User can set his own macro economy forecast by calling set_pa_custom_scenario().

```
*  ...
*    open_deal_ex(tid, pCmo);
*    set_moodys_credit_model_settings(tid,
      MOODYS_PA_SETTINGS, false);
*    // set custom PA mode
*    set_pa_analysis_type(tid, PA_CUST_MEDC_SINGLE_PATH);
*
*    int year[]={2013,2014,2014,2014,2014,2015,2015,2015,2015,2016,2016,2016,2016,2017,2017,2017,2017,2018,
      2018,2018};
*    int quarter[]={4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3};
*    double gdp[] = {0.99,1.34,1.77,1.98,2.09,2.43,2.53,2.69,3.16,3.87,4.05,5.11,6.19,9.67,13.42,15.55,16.9
      5,18.34,23.39,29.66};
*    //set custom GDP forecast
*    set_pa_custom_scenario(tid, "GDP", year, quarter, gdp, 20);
*
*    run_deal_ex(tid, pCmo);
*    // get cashflow result code here
*  ...
*
```

Chapter 8

Stress EDF Credit Model

8.1 Overview

Stress EDF credit model is composes 2 parts:

- EDF(expected default frequency) data for each collateral.
- Algorithm to project default assumption base on each collaterals' expected default frequency.

To project cashflow base on Stress EDF credit model, user need to :

1. Set EDF data for each collateral:
   - Set EDF data by applying predefined scenario.
   - Set EDF data base on user's own judgement.
2. Call run_deal_ex(), API will automatically apply algorithm to generate default assumption base on collateral's EDF data.

8.2 Running a Stress EDF

8.2.1 Step 1: Prerequisite

- WSA API version 2.1 or newer.
- EDF database file(DEAL_NAME_EDF.DB) exists in the deal file.(The database file is required for Stress EDF Scenario)
- Calling set_moodys_credit_model_settings() with MOODYS_SEDF_SETTINGS.

8.2.1.1 Select Stress EDF credit model

The very first step to use Stress EDF credit model is calling set_moodys_credit_model() with enumeration MOODYS_SEDF_SETTINGS :

```
*   ...
*    open_deal_ex(tid, pCmo);
*    set_moodys_credit_model_settings(tid,
       MOODYS_SEDF_SETTINGS, 1);
*   ...
*
```

Warning

> if the parameter sets_up_only in set_moodys_credit_model_settings() set to 0, API will set prepay assumption with 30 CPR and recovery lag with 15.
> These prepay/recovery assumption won't be overrided by either set_prepayments_ex() or set_recovery_lag_-ex()

8.2.1.2 Settlement date

If user open a deal with a settlement date at CMO_STRUCT::settlement_date, like:

```
*  ...
*   char deal_id[9] = "JUBILEE4";
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   pCmo->actual_coll = 1;
*   strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*   strncpy_s(pCmo->settlement_date, "06/15/2014", 10);
*   open_deal_ex(tid, pCmo);
*  ...
*
```

API will load EDF data as of June,2014 from local EDF database after calling set_moodys_credit_model_settings().

If no EDF data available for settlement date, API will pick nearest available EDF data.

If user doesn't specify settlement date when he opens the deal, API will load latest available EDF data for Stress EDF run.

8.2.2 Step 2 : Setting Stress EDF data

There are 2 sources setting Stress EDF data, one is loaded from local Stress EDF database, the other is set custom EDF data.

8.2.2.1 Run Stress EDF with predefined scenario

API provide a list of pre-defined macro economy scenarios for Stress EDF. User can select one of the scenarios to run Stress EDF mode.

Base on different economy scenario, each collateral will be apply with different default probabilities. Stress EDF will project collateral default principals base on these default probabilities.

Get scenario list

To get full list of scenarios avaiable in current EDF database, user need to call get_edf_scenarios().

1. Get the number of scenarios avaialble

```
*  ...
*   // make sure .SFW deal file has EDF database.
*   // get the number of all avaible EDF scenarios in database.
*   int num_edf_scenarios = get_edf_scenarios(tid, NULL);
*   printf("%d scenarios found \n", num_edf_scenarios);
*  ...
*
```

2. Populate scneario names

```
*  ...
*   char * scenario_list[9];
*   for(int i=0;i<num_edf_scenarios;i++){
*        scenario_list[i] = (char *)malloc(sizeof(char) * 20);
*   }
*
*   for(int i=0;i<num_edf_scenarios;i++){
*        printf("name of scenario %d: %s \n", i,scenario_list[i]);
*   }
*  ...
*
```

Set/view current scenario

```
*   ...
*     // set scenario 3, 3 is the index of scenarios return by get_edf_scenarios()
*     set_current_edf_scenario(tid, 3);
*   ...
*
```

View EDF(expected default frequency) data

After user select scenario by calling set_current_edf_scenario(), API will apply default probabilities to collaterals.

User can inspect these default value by calling get_loan_edf()

```
*   ...
*    run_deal_ex(tid,pCmo)
*    // create an array to hold collateral edf data
*    double edf_loan_data[5] = {0.0};
*    for(int i =1;i<=pCmo->num_colls;i++){
*        // popluate collateral loan edf data
*        get_loan_edf(tid, NULL,i,edf_loan_data,5);
*        printf("Coll_num:%d, default probability in year 1:%f,  default probability in year 2:%f \n", i,
      edf_loan_data[0], edf_loan_data[1]);
*    }
*   ...
*
```

8.2.2.2 Run Stress EDF with customized EDF data

API also provide function "set_loan_edf()" enable user to set EDF data base on their own judgenment.

```
*   ...
*    double edf_loan_data[5] = {0.0};
*    for(int i =1;i<=pCmo->num_colls;i++){
*        get_loan_edf(tid, NULL,i,edf_loan_data,5);
*        printf("Coll_num:%d, dp in year 4:%f,  dp in year 5:%f \n", i,edf_loan_data[3], edf_loan_data[4]);
*           //scale up 20% for default probability in year 5
*        edf_loan_data[4] *= 1.2;
*           //scale up 10% for default probability in year 4
*        edf_loan_data[3] *= 1.1;
*           //set modified EDF data back to API
*        set_loan_edf(tid, NULL, i, edf_loan_data, 5);
*           //check the EDF data set
*        get_loan_edf(tid, NULL, i, edf_loan_data, 5);
*        printf("Coll_num:%d, dp in year 4:%f,  dp in year 5:%f \n", i,edf_loan_data[3], edf_loan_data[4]);
*    }
*   ...
*
```

8.2.3 Step 3 : Applying user assumption

Warning

> user assumption can only be applied if user call set_moodys_credit_model_settings() with sets_up_only=1. If set_moodys_credit_model_settings() called with sets_up_only=0. API will use hard-coded scenario -: Prepayment: 30% CPR and 15 months recovery lag.

Set a prepayment

When running with Stress EDF credit model, user are also able to apply prepayment assumption on deal level. Please refer to set_prepayments_ex().

Set a default multiplier

User can scale up or scale down the default assumption by calling set_edf_default_multiplier().

```
*  ...
*      // this will scale up defaults to 5% higher than original Stress EDF defaults amount.
*      set_edf_default_multiplier(tid, 1.05);
*  ...
*
```

8.2.4 Step 4 : Getting cashflow result

The way getting the cashflow after running Stress EDF is the same way as running a normal deal in WSA API.

Chapter 9

CMM Credit Model

9.1 Overview

The CMM (Commercial Mortgage Metrics) model is the leading analytical tool for assessing default and recovery risk for commercial real estate loans. It enables you to perform stress tests and determine loan loss provisions on your commercial mortgage portfolios.

9.2 Running CMM

9.2.1 Step 1: Select CMM credit model

The very first step to use CMM credit model is calling set_moodys_credit_model() with enumeration MOODYS_C-MM_SETTINGS :

```
*  ...
*   open_deal_ex(tid, pCmo);
*   set_moodys_credit_model_settings(tid,
      MOODYS_CMM_SETTINGS, 1);
*  ...
*
```

9.2.2 Step 2: View/set CMM predefined scenarios

API can get a list of available CMM pre-defined economy scenarios, which is read from deal file. User can call API to know which secnario is active currently. User also can select one of the scenarios, make it active to run CMM.

Base on different economy scenario, each loan of collateral will be applied with different default probabilities and LGD. CMM will project collateral default principals base on these default probabilities.

And it will project collateral recovery principals base on these LGD values.

Get scenario list

To get full list of scenarios avaiable in current deal, user need to call get_moodys_cmm_scenarios().

1. Get the number of scenarios avaialble

```
*  ...
*   // make sure .SFW deal file has CMM related files.
*   // get the number of all avaible CMM scenarios in current deal.
*   int num_cmm_scenarios = get_moodys_cmm_scenarios(tid, NULL, NULL);
*   printf("%d scenarios found \n", num_cmm_scenarios);
*  ...
*
```

2. Populate scneario names

```
*   ...
*     char **scenarios = new char*[num_cmm_scenarios];
*     for (int i = 0; i < num_cmm_scenarios; ++i)
*       scenarios[i] = new char[20];
*     get_moodys_cmm_scenarios(tid, NULL, scenarios);
*
*    for(int i=0;i<num_cmm_scenarios;i++){
*        printf("name of scenario %d: %s \n", i,scenarios[i]);
*    }
*   ...
*
```

View current cmm scenario

```
*   ...
*     char current_scenario[20];
*     get_current_moodys_cmm_scenario(tid, NULL, current_scenario);
*     printf("current active CMM scenario is %s \n", current_scenario);
*   ...
*
```

Set current cmm scenario

```
*   ...
*     char **scenarios = new char*[num_cmm_scenarios];
*     for (int i = 0; i < num_cmm_scenarios; ++i)
*       scenarios[i] = new char[20];
*     get_moodys_cmm_scenarios(tid, NULL, scenarios);
*     set_current_moodys_cmm_scenario(tid, NULL, scenarios[2]);
*   ...
*
```

9.2.3 Step 3: Run and get cashflow result

After call run_deal_ex(), API will start to run CMM credit model. The way getting the cashflow after running CMM is the same way as running a normal deal in WSA API.

Chapter 10

Default Probability Distribution

10.1 Overview

Default Probability Distribution Simulation generates paths of cashflows using user defined default/loss scenarios and default timing. The simulation provide bond ratings based on the cashflows.

10.2 Running Default Probability Distribution

10.2.1 Step 1 :Set Simulation Engine

To run default probability distribution simulation, users should first explicitly specify the simulation engine by calling set_simulation_engine() with argument SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION

```
*  ...
*     open_deal_ex(pDeal, pCmo);
*     // call set_simulation_engine with SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION
*     set_simulation_engine(
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION);
*  ...
*
```

10.2.2 Step 2 :Set DPD assumption

10.2.2.1 Set DPD_ASSUMPTION structure

After setting the simulation engine, users should set up simulation assumptions by creating DPD_ASSUMPTION and configure on it.

Then user calls set_dpd_assumption() to set assumptions.

```
*  ...
*    set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION);
*
*    DPD_ASSUMPTION dpdAssumption;
*    // one of enum DPD_DISTRIBUTION_TYPE
*    dpdAssumption.distribution = DPD_DISTRIBUTION_LOGNORMAL;
*    // scenario type, 0 for defaults and 1 for losses
*    dpdAssumption.scenario_type = 1;
*    // mean for the probability distribution, required when the distribution type is lognormal or inverse
       normal
*    dpdAssumption.mean = 10.0;
*    // standard deviation for the probability distribution, required when the distribution type is
       lognormal or inverse normal
*    dpdAssumption.standard_deviation = 1.5;
*    // If TRUE then use the Milan Aaa CE input, use Standard Deviation if otherwise.
*    dpdAssumption.use_milan_aaa_ce = 0;
*    // Milan Aaa CE for generating distributions if distribution type is lognormal and scenario type is
```

```
      losses.
*    dpdAssumption.milan_aaa_ce = 0.0;
*    // when the scenario type is losses,
*    dpdAssumption.discounted_recoveries = 0;
*    // revolving default factor
*    for(int i=0;i<500;i++)
*          dpdAssumption.revolving_default_factors[i] = 0.01;
*    // number of scenarios for the simulation to run.
*    dpdAssumption.num_scenarios = 3;
*    // TRUE is users want to use separate default timing curve for revolving assets.
*    dpdAssumption.use_revolving_def_timing = 1;
*    // primary proposed rating cap
*    dpdAssumption.rating_cap_primary = 3;
*    // surveillance proposed rating cap
*    dpdAssumption.rating_cap_surveillance = 5;
*
*    set_dpd_assumption(tid, &dpdAssumption);
*
*    run_default_probability_distribution(pDeal);
*  ...
*
```

10.2.2.2 Set Default Timing

Users should also use methods set_dpd_current_default_timing() and set_dpd_revolving_default_timing() to setup the default timing vectors.

```
*  ...
*      set_simulation_engine(pDeal,
     SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION);
*
*      double currentDefaultTimingVector[] = {0.01};
*      set_dpd_current_default_timing(pDeal, currentDefaultTimingVector, 1, 0
     );
*      double revolvingDefaultTimingVector[] = {0.02};
*      set_dpd_revolving_default_timing(pDeal, revolvingDefaultTimingVector
     , 1, 0);
*
*      run_default_probability_distribution(pDeal);
*  ...
*
```

Both vectors can be retrieved by using method get_dpd_current_default_timing() and get_dpd_revolving_default_-timing() (if DPD_ASSUMPTION.use_revolving_def_timing set to 1).

10.2.2.3 Use User Defined Distribution

if users choose to use user defined distribution, he also need to setup the scenarios using set_dpd_scenarios() by giving a scenario vector. Structure DPD_SCENARIO has following fields.

```
*  ...
*    set_simulation_engine(pDeal,
     SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION);
*  ...
*    // make sure assign dpdAssumption.distribution with DPD_DISTRIBUTION_USER_DEFINED
*    dpdAssumption.distribution = DPD_DISTRIBUTION_USER_DEFINED ;
*  ...
*    DPD_SCENARIO userScenarios[] = {
*            {1, 6.00, 8.00},
*            {2, 11.00, 90.00},
*            {3, 13.00, 1.0},
*            {4, 16.00, 1.0}
*        };
*       set_dpd_scenarios(pDeal, userScenarios, sizeof(userScenarios)/sizeof(userScenarios
     [0]));
*  ...
*
*       double currentDefaultTimingVector[] = {0.01};
*       set_dpd_current_default_timing(pDeal, currentDefaultTimingVector, 1,
     0);
*
*    run_default_probability_distribution(pDeal);
*  ...
*
```

After setting up the assumption, users can use method get_dpd_scenarios() to view the scenario vector.

10.2.3 Step 3 :Run the Simulation

After setting up the simulation assumption, users can run the DPD Simulation by calling function run_default_-probability_distribution_simulation()

```
*  ...
*      run_default_probability_distribution(pDeal);
*  ...
*
```

10.2.4 Step 4 : Getting Results

10.2.4.1 Getting Cashflow Result

Simulation cashflow results can be retrieved using following functions after run.

- get_collateral_flow_sim() - users can retrieve collateral flow of a specific path by giving the path number. Average collateral flow can be retrieved by giving 0 for the path number.

```
*  ...
*      run_default_probability_distribution(pDeal));
*      // get losses cashflows from 1st path in the simulation
*      double* losses = get_collateral_flow_sim(pDeal, 1,
      FLOW_COLLATERAL_LOSSES);
*  ...
*
```

- get_bond_flow_sim() - users can retrieve cashflow of a specific bond of a specific path by giving the bond id and path number. Average bond flow can be retrieved by giving 0 for path number.

```
*  ...
*      run_default_probability_distribution(pDeal));
*      // get balances of Bond "A" from 1st path in the simulation
*      double* balance = get_bond_flow_sim(pDeal, 1, "A1",
      FLOW_BOND_BALANCE);
*  ...
*
```

10.2.4.2 Getting Bond Performance Result

- get_dpd_results() - users can retrieve bond performance statistics by giving bond id.

```
*  ...
*    run_default_probability_distribution(pDeal);
*
*    DPD_RESULT * pDPD_Result = &DPD_RESULT();
*    get_dpd_results(pDeal,"A1",pDPD_Result);
*    //inspect values from result structure
*    printf(pDPD_Result->balance);
*    close_deal_ex(pDeal, pCmo);
*  ...
*
```

Chapter 11

Monte Carlo Simulation

11.1 Overview

Monte Carlo simulation generates random paths by simulating asset default times using user inputs.

Using corresponding syntax, users can retrieve bond/collateral cash flows of every individual path and performing statistics.

11.2 Running Monte Carlo Simulation

11.2.1 Step 1 :Set Simulation Engine

To run Monte Carlo simulation, users should first explicitly specify the simulation engine by calling set_simulation_-engine() with argument SIMULATION_MONTE_CARLO

```
*  ...
*     open_deal_ex(pDeal, pCmo);
*     // call set_simulation_engine with SIMULATION_MONTE_CARLO
*     set_simulation_engine(SIMULATION_MONTE_CARLO);
*  ...
*
```

11.2.2 Step 2 :Set Monte Carlo assumptions

11.2.2.1 "Auto" Mode

"Auto" mode generates random paths by simulating asset default times, After setting the simulation engine, to run in “Auto” mode, users should set up simulation assumptions by calling set_monte_carlo_assumption().

Users need to fill information in two structures MONTE_CARLO_ASSUMPTION and MONTE_CARLO_DEF_PPY-_REC_ASSUMPTION.

Warning

The fields that are required to run Monte Carlo simulation on ABS and CLO are slightly different.

MONTE_CARLO_ASSUMPTION

```
*  ...
*     set_simulation_engine(pDeal, SIMULATION_MONTE_CARLO);
*
*     MONTE_CARLO_ASSUMPTION basic_assumption;
*     // assign members of basic_assumption
*     // 0 for Auto Mode, 1 for Input Mode
*     basic_assumption.mode = 1 ;
*     // The number of paths to be simulated
*     basic_assumption.num_paths = 50 ;
*     // specifies if users want to run with optimization.
*     // MC_OPTIMIZATION_NONE for no optimization
*     // MC_OPTIMIZATION_PATHS for path optimization
*     // MC_OPTIMIZATION_TAIL_RUN for tail optimization
*     basic_assumption.optimization_type = MC_OPTIMIZATION_NONE;
*     // optimization_pct is required for run with optimization
*     basic_assumption.optimization_pct = 0.3;
*     // ONLY for CLO. Indicating if reinvestment should be modelled using "CURRENT" or "REINV" pool.
*     // 0 for "REINV" and 1 for "CURRENT"
*     basic_assumption.reinv_pool = 0 ;
*     // ONLY for CLO to indicate if apply default to "REINV" pool.
*     basic_assumption.default_to_reinv = 0 ;
*
*     MONTE_CARLO_DEF_PPY_REC_ASSUMPTION def_ppy_rec_assumption;
*     // assign members of def_ppy_rec_assumption
*     set_monte_carlo_assumption(pDeal, &basic_assumption, &
      def_ppy_rec_assumption);
*   ...
*
```

MONTE_CARLO_DEF_PPY_REC_ASSUMPTION

```
*   ...
*    set_simulation_engine(pDeal, SIMULATION_MONTE_CARLO);
*
*    MONTE_CARLO_ASSUMPTION basic_assumption;
*    // assign members of basic_assumption
*
*    MONTE_CARLO_DEF_PPY_REC_ASSUMPTION def_ppy_rec_assumption;
*    // 0 for "Industry Correlation", 1 for "Portfolio Correlation" and 2 for "Global Correlation(SFW)".
*    def_ppy_rec_assumption.correlation_type = 1;
*    // copula function, 0 for Gaussian and 1 for Student-T.
*    def_ppy_rec_assumption.copula = 0 ;
*
*    set_monte_carlo_assumption(pDeal, &basic_assumption, &def_ppy_rec_assumption
      );
*    run_monte_carlo_simulation(pDeal);
*   ...
*
```

11.2.2.2 Loan level cash flow

"Input" mode generate paths using user input default times.

To run in "Input" mode, users should provide default timing and recovery rate for every asset by using method set_monte_carlo_default_time_and_recovery().

```
*   ...
*    set_simulation_engine(pDeal, SIMULATION_MONTE_CARLO);
*    //In Path 1, Loan with index = 0 will default at period 4 with recovery rate 20%
*    set_monte_carlo_default_time_and_recovery(pDeal, 1, 0, 4,0.2)
      ;
*    run_monte_carlo_simulation(pDeal);
*   ...
*
```

11.2.2.3 Retrieve pricing results

Default

When running ABS or running CLO using asset default probability, users should setup EDF data for the loans using set_loan_edf().

Recovery

For ABS, users should use set_loan_lgd() to setup loan LGD.

For CLO/CDO, users should use set_recoveries_ex() to set asset recoveries.

For ABS/CLO/CDO, users can also use set_recovery_lag_ex() to set the recovery lag for the assets

11.2.3 Step 3 :Run the Simulation

After setting up the simulation assumption, users can run the Monte Carlo Simulation by calling function run_monte-_carlo_simulation()

```
*  ...
*      run_monte_carlo_simulation(pDeal);
*  ...
*
```

11.2.4 Step 4 : Getting Results

Simulation results can be retrieved using following methods after run.

- get_collateral_flow_sim() - users can retrieve collateral flow of a specific path by giving the path number. Average collateral flow can be retrieved by giving 0 for the path number.

```
*  ...
*      run_default_probability_distribution(pDeal));
*      // get losses cashflows from 1st path in the simulation
*      double* losses = get_collateral_flow_sim(pDeal, 1,
      FLOW_COLLATERAL_LOSSES);
*  ...
*
```

- get_bond_flow_sim() - users can retrieve cashflow of a specific bond of a specific path by giving the bond id and path number. Average bond flow can be retrieved by giving 0 for path number.

```
*  ...
*      run_default_probability_distribution(pDeal));
*      // get balances of Bond "A" from 1st path in the simulation
*      double* balance = get_bond_flow_sim(pDeal, 1, "A1",
      FLOW_BOND_BALANCE);
*  ...
*
```

- get_monte_carlo_result() - users can retrieve bond performance statistics by giving bond id.

```
*  ...
*    run_monte_carlo_simulation(pDeal);
*
*    MONTE_CARLO_RESULT result;
*    get_monte_carlo_result(pDeal,"A1",&result);
*    //inspect values from result structure
*    printf(result->total_principal);
*    close_deal_ex(pDeal, pCmo);
*  ...
*
```

Chapter 12

Whole Loan Analyzer

12.1 Overview

Since WSAAPI v2.7, the whole loan analysis was introduced - Whole Loan Analyzer.

By setting up loans in WSAAPI Whole Loan Analyzer, users can generate loan & portfolio level cash flow, run credit model and price individual loans.

Whole Loan Analyzer is powered by SFW engine, which is for most mortgage types, or CDONet engine, which is for bankloans.

12.2 Steps for Whole Loan Analysis

12.2.1 Set whole loan

By calling set_whole_loan(), loan tapes could be loaded into engine.

Each valid loan would be allocated with a "loan_number" sequentially by the order in which the loans are supplied to the engine, starting from "1".

```
*  ...
*    void* tid(NULL);
*
*    // Set the path for "WSA_API.DB" and whole loan LIC.
*    const char input_path[] = "..\\Data\\";
*    set_input_path(input_path);
*
*    // Set the preferred engine for overlapping deals.
*    set_engine_preference(PICK_SFW_ENGINE_FOR_MAPPED_DEALS
      );
*    // Clean up temporary files
*    set_tmp_dir_treatment(TMP_DIR_REMOVE_ALL);
*
*    // Set the preferred engine for overlapping deals.
*    set_engine_preference(PICK_SFW_ENGINE_FOR_MAPPED_DEALS
      );
*    vector<WHOLE_LOAN_STRUCT> loans;
*
*    WHOLE_LOAN_STRUCT loan;
*    memset(&loan, 0, sizeof(loan));
*
*    loan.amortization_type = ANN;  //amortization type
*
*    loan.current_balance = 10000;
*    loan.term = 12;
*    loan.original_term = 360;
*    strncpy_s(loan.currency, "USD", 4);
*
*    coupon info
*    COUPON_INFO& coupon_info = loan.coupon_info;
*    coupon_info.coupon_type =
*    WHOLE_LOAN_COUPON_TYPE::FIXED_COUPON;
*    coupon_info.gross_coupon = 0.05;
*    coupon_info.net_coupon = 0.05;
*    coupon_info.day_count = DAYCOUNT_30_360;
*    coupon_info.pay_freq = PAYMENT_FREQUENCY::PAY_FREQ_MONTHLY;
*
*    loans.push_back(loan);
*
*    const int settlementdate = 20160201;
*    set_whole_loan(tid, &loans.front(), loans.size(), settlementdate);
*
*    // set calc level to enable loan cashflow
*    set_deal_calc_level(tid, CALC_LEVEL_FULL_WITH_LOAN, 1);
*
*    CMO_STRUCT *pCmo = new CMO_STRUCT();
*    run_deal_ex(tid, pCmo);
*  ...
*
```

12.2.2 Get cash flow and pricing results

12.2.2.1 Portfolio cash flow

Retrieving portfolio level cashflow in Whole Loan Analyzer is pretty similar to the way of running regular deals. Please refer category index 12.1.1 - "Collateral cashflow" for details.

12.2.2.2 Loan level cash flow

There are 2 ways to retrieve loan level cashflow - either by calling get_loan_flow_ex() to populate structure MOODYS_LOAN_CASHFLOW or by calling get_loan_flow() to get a single flow.

Loan level cashflow is only retrievable when deal_calc_level is set to CALC_LEVEL_FULL_WITH_LOAN.

get_loan_flow_ex():

get_loan_flow_ex() is recommended to be used if users would like to get a set of a loan's cashflow by calling API function once.

```
*  ...
*    // set calc level to enable loan cashflow
*    set_deal_calc_level(tid, CALC_LEVEL_FULL_WITH_LOAN, 1);
*
*    CMO_STRUCT *pCmo = new CMO_STRUCT();
*    run_deal_ex(tid, pCmo);
*
*    MOODYS_LOAN_CASHFLOW * pMLC = new MOODYS_LOAN_CASHFLOW();
*    get_loan_flow_ex(tid, 1, pMLC);
*    for (int j = 0;j < pMLC->size;j++)
*    {
*        printf("%d,%f\n", pMLC->dates[j], pMLC->balance[j]);
*    }
*  ...
*
```

get_loan_flow():

MOODYS_LOAN_CASHFLOW only provides a limit number of cashflows. WSAAPI Whole Loan Analyzer also provides get_loan_flow() to offer more loan flows.

get_loan_flow() is recommended to be used if users would like to retrieve cashflows that wasn't included in MOODYS_LOAN_CASHFLOW.

Note: Full list of available identifier (some may not available in Whole Loan Analyzer): EXTENDED_FLOW_COLLATERAL_IDENTIFIER

```
*  ...
*    // set calc level to enable loan cashflow
*    set_deal_calc_level(tid, CALC_LEVEL_FULL_WITH_LOAN, 1);
*
*    CMO_STRUCT *pCmo = new CMO_STRUCT();
*    run_deal_ex(tid, pCmo);
*
*    double* loan_cf_balance = get_loan_flow(tid, 1, NULL,
      FLOW_COLLATERAL_BALANCE);
```

```
*    double* loan_cf_cash = get_loan_flow(tid, 1, NULL,
      FLOW_COLLATERAL_CASH);
*  ...
*
```

12.2.2.3 Retrieve pricing results

WSAAPI Whole Loan Analyzer supports cashflow analytics for a given loan.

Loan pricing functionality is only available after running cashflow, with deal_calc_level being set to CALC_LEVEL_-FULL_WITH_LOAN.

```
*  ...
*    // set calc level to enable loan cashflow
*    set_deal_calc_level(tid, CALC_LEVEL_FULL_WITH_LOAN, 1);
*
*    CMO_STRUCT *pCmo = new CMO_STRUCT();
*    run_deal_ex(tid, pCmo);
*
*    PRICING_RESULTS* pr = new PRICING_RESULTS();
*    price_loan(tid, 1, PRICE, 100, pr);
*  ...
*
```

Chapter 13

Market Risk Metrics

13.1 Overview

Since API 3.3.0, the Market Risk Metrics module has been enabled for all asset types and bank loans, which provides static and option-adjusted market risk measures through independent function calls.

13.2 Static Market Risk Metrics

Available metrics are: i-spread, z-spread, DV01, Macaulay duration, effective yield, yield-to-worst, yield value of the 32nd, spread convexity and the annualized measures.

13.2.1 Step1: Set Reference Index Rate

To calculate spreads and spread related metrics such as i-spread and z-spread, user is required to set the reference curve(s) by calling set_index_rate() or equivalent functions. For example:

```
*  ...
*    // open deal
*    open_deal_ex(pDeal, pCmo);
*
*    // call set_index_rate to set reference index rate
*    short index = LIBOR_3;
*    double rate = 0.028;
*    set_index_rate(pDeal, "USD", &index, 0, &rate);
*  ...
*
```

13.2.2 Step2: Set Static Market Risk Metrics Input

Besides the reference rate curves, user may refer to structure METRIC_INPUT_STRUCT which stores inputs for static metrics calculation.

User may set clean_price value and APPLY_SPREAD_TYPE(LIBOR or TSY) as follows:

```
*  ...
*    METRIC_INPUT_STRUCT  metric_input;
*
*    metric_input.clean_price = 100.0;
*    metric_input.apply_spread_to = APPLY_SPREAD_TO_LIBOR;
*  ...
*
```

13.2.3 Step3: Get Static Risk Measures Result

METRIC_RESULTS_STRUCT stores static risk metrics that can be retrieved by calling get_bond_market_risk_-metrics() or get_loan_market_risk_metrics() after the deal is run.

```
*  ...
*    run_deal_ex(tid, pCmo);
*
*    METRIC_RESULTS_STRUCT results_m;
*    get_bond_market_risk_metrics(tid, bondId, &metric_input, &results_m);
*    close_deal_ex(pDeal, pCmo);
*  ...
*
```

13.3 Option-Adjusted Market Risk Metrics

Available metrics are: option-adjusted spread (OAS), effective duration spot, effective duration par, effective convexity spot, effective convexity par, spread duration.

13.3.1 Pre-Step1 for Bankloan only: Set bank loan call model parameters

Bank Loan Call Model was developed and implemented to simulate a bank loan's call behavior with integration of ESG credit spread simulation model.

Certain parameters are needed to be set by calling set_bankloan_call_adj_param(). WSAAPI provides recommended settings for bankloans.

13.3.2 Pre-Step2 for Bankloan only: Set simulated credit spreads

Credit Spread simulation of different credit ratings & terms is necessary for bank loan call model. User-specified simulated bankloan credit spreads can be set by calling set_spot_spread().

WSAAPI is also integrated with ESG's credit spread simulation, which needs to be triggered by calling set_up_ESG_model_interest_rates() with ESG_MODEL_INPUTS::ESG_RATE_TYPE being set to SPOT_SPREAD_RATE.

WSAAPI can provide the necessary implementation logic and adjustment for bankloans.

13.3.3 Step1: Set ESG simulated index rates (optional)

API is integrated with ESG's extended 2-Factor Black-Karasinski Model (WSAESGProvider.dll), which provides simulated LIBOR/SWAP and government curves of over 30 economies.

User may set up this dynamic simulation process by specifying structure ESG_MODEL_INPUTS and calling function set_up_ESG_model_interest_rates().

Alternatively, user may set the simulated interest rate paths generated by his/her own interest rate model as described in Step3 below.

13.3.4 Step2: Set OAS-related Market Risk Metrics Input

METRIC_INPUT_STRUCT_EX is the structure to hold inputs specifically for OAS-related analysis.

User may specify simulation path number, OAS mode and interest rate shift amount using this structure, and call function set_metrics_input_ex() to set the inputs into API.

```
*  ...
*    METRIC_INPUT_STRUCT_EX metric_input_ex;
*
*    // 100 paths for OAS analysis
*    metric_input_ex.num_paths = 100;
*     // enable calculation of all OAS-related metrics
*     metric_input_ex.oas_mode = ENABLE_ALL;
*     // shift the reference curve up (down) by 100 basis points
*     metric_input_ex.shift_amt = 0.01;
*
*     set_metrics_input_ex(tid, &metric_input_ex);
*   ...
*
```

13.3.5 Step3: Apply user-specified simulated interest rates (optional)

User may choose to set his/her own interest rate paths for OAS analysis by calling function set_index_rate_ex() after Step2.

If neither Step1 nor Step3 is executed, the static interest rate path from function call set_index_rate() or equivalent would be applied to all paths for OAS analysis.

13.3.6 Step4: Get OAS-related Risk Measures Result

METRIC_RESULTS_STRUCT_EX stores results of OAS related simulation risk measures, which can be retrieved by calling get_bond_market_risk_metrics_ex() or get_loan_market_risk_metrics_ex().

In addition to the inputs detailed in the above section that generally apply to the entire waterfall model, user is required to set three more parameters for the requested tranche or bankloan:

APPLY_SPREAD_TYPE, METRIC_ANCHORS (OAS or MARKET_PRICE) and anchor_value.

```
*   ...
*     run_deal_ex(tid, pCmo);
*
*     METRIC_RESULTS_STRUCT_EX results_ex;
*     get_bond_market_risk_metrics_ex(tid, bondId,
       MARKET_PRICE, 100.0, APPLY_SPREAD_TO_LIBOR, &results_ex);
*     // get_loan_market_risk_metrics_ex(tid, loan_num, MARKET_PRICE, 100.0, APPLY_SPREAD_TO_LIBOR,
        &results_ex); //for bankloans
*     close_deal_ex(pDeal, pCmo);
*   ...
*
```

Chapter 14

How To

14.1 How to get cashflows result

After a deal run by calling run_deal_ex(), user are able to get both collateral cashflows and bond cashflows.

14.1.1 Collateral cashflow

There are 2 ways to retrieve collateral cashflows.

Either by calling get_collateral_flow_ex() to get a single flows or by calling get_collateral_flow_ex1() to popluate structure MARKIT_COLLAT_CASHFLOW.

Collateral Groups

A deal may has more than one group of collaterals. API provide function view_coll_groups() to get all group numbers.

```
*  ...
*   //get total number of collateral groups
*   int num_of_groups = 0 ;
*   view_coll_groups(tid,NULL, NULL,&num_of_groups);
*   printf("Number of collateral groups: %d \n", num_of_groups);
*
*   //get group number in an array
*   int * groups_array;
*   groups_array = (int*)malloc(sizeof(int)*num_of_groups);
*   //popluate groups_array by calling view_coll_groups()
*   view_coll_groups(tid, groups_array ,num_of_groups, &num_of_groups);
*   for (int i=0;i<num_of_groups;i++)
*       printf("Group Number: %d \n", groups_array[i]);
*  ...
*
```

get_collateral_flow_ex1()

User should use get_collateral_flow_ex1() if he wants to :

- Getting a set of collateral cashflows by calling API function once
- Getting underlying deal's collateral cashflow

if group_number = 0, API will popluate MARKIT_COLLAT_CASHFLOW with deal level collateral cashflow if reremic-_deal_id_or_null = NULL, API will popluate MARKIT_COLLAT_CASHFLOW with current(top deal) collateral cash-flow

```
*  ...
*   run_deal_ex(tid, pCmo);
*      //new a MARKIT_COLLAT_CASHFLOW to hold collateral flow
*   MARKIT_COLLAT_CASHFLOW *mcc = new
      MARKIT_COLLAT_CASHFLOW();
*      //popluate MARKIT_COLLAT_CASHFLOW by calling get_collateral_flow_ex1()
*   get_collateral_flow_ex1(tid, 0, NULL, mcc);
*      //inspect collateral cashflow result
*   for(int i=0;i<mcc->size;i++)
*       printf("date:%d, balance:%f, default:%f \n",mcc->dates[i], mcc->
      balance[i],mcc->defaults[i]);
*  ...
*
```

Get collateral cashflow from underlying deal

For example, there is a top deal with deal_id "CML12004-I" and it has an underlying deal with deal_id "CHL06HB1".

```
*  ...
*   run_deal_ex(tid, pCmo);
*   MARKIT_COLLAT_CASHFLOW *mcc = new
      MARKIT_COLLAT_CASHFLOW();
*   // this will populate deal level collateral cashflow from underlying deal "CHL06HB1"
*   get_collateral_flow_ex1(tid, 0, "CHL06HB1", mcc);
*   for(int i=0;i<mcc->size;i++)
*       printf("date:%d, balance:%f, default:%f \n",mcc->dates[i], mcc->
      balance[i],mcc->defaults[i]);
*  ...
*
```

get_collateral_flow_ex()

MARKIT_COLLAT_CASHFLOW only provides a limit number of collateral flows. API provide get_collateral_flow_-ex() to offer more collateral flows.

User should use this function if he want to inspect cashflows that wasn't included in MARKIT_COLLAT_CASHFLO-W.

Note

Full list of available identifier : EXTENDED_FLOW_COLLATERAL_IDENTIFIER

```
*  ...
*   run_deal_ex(tid, pCmo);
*
*   double * collateral_balance;
*   double * coll_balance_gp1;
*   double * coll_balance_gp2;
*   double * coll_balance_gp3;
*   double * coll_balance_gp4;
*   collateral_balance = get_collateral_flow_ex(tid, 0,
      FLOW_COLLATERAL_BALANCE);
*   //get balance flow from group 1
*   coll_balance_gp1 = get_collateral_flow_ex(tid, 1,
      FLOW_COLLATERAL_BALANCE);
*   //get balance flow from group 2
*   coll_balance_gp2 = get_collateral_flow_ex(tid, 2,
      FLOW_COLLATERAL_BALANCE);
*   //get balance flow from group 3
*   coll_balance_gp3 = get_collateral_flow_ex(tid, 3,
      FLOW_COLLATERAL_BALANCE);
*   //get balance flow from group 4
*   coll_balance_gp4 = get_collateral_flow_ex(tid, 4,
      FLOW_COLLATERAL_BALANCE);
*   for (int i =0;i<MAX_PERIODS;i++){
*       printf("Collateral Balance(0): %f", collateral_balance[i]);
*       printf("Collateral Balance(1): %f", coll_balance_gp1[i]);
*       printf("Collateral Balance(2): %f", coll_balance_gp2[i]);
*       printf("Collateral Balance(3): %f", coll_balance_gp3[i]);
*       printf("Collateral Balance(4): %f \n", coll_balance_gp4[i]);
*   }
*  ...
*
```

14.1.2 Bond cashflow

Similar with retriving collateral cashflow, API provide two functions call to get bond cashflow get_bond_flow_ex() and get_bond_flow_ex1().

get_bond_flow_ex1()

User should use get_bond_flow_ex1():

- Getting a set of bond cashflows by calling API function once
- Getting underlying deal's bond cashflow

Note

User should pass reremic_deal_id_or_null with NULL if the bondid indicating the bond in parent deal.
User should pass reremic_deal_id_or_null with deal id if the bondid indicating the bond in underlying deal.

```
*   ...
*   run_deal_ex(tid, pCmo);
*   // new MARKIT_BOND_CASHFLOW to hold bond cashflow result
*   MARKIT_BOND_CASHFLOW * pMBC = new MARKIT_BOND_CASHFLOW();
*   // return pointer to MARKIT_BOND_CASHFLOW structure with bond cashflow populated
*   pMBC = get_bond_flow_ex1(tid, NULL, "1A24");
*   // inspect the bond cashflow
*   for(int i=0;i<pMBC->size;i++)
*       printf("Balance:%f,Principal:%f,Interest:%f \n",pMBC->balance[i],pMBC->
     principal[i],pMBC->interest[i]);
*   ...
*
```

get_bond_flow_ex()

User should use get_bond_flow_ex() if he want to inspect cashflows that wasn't included in MARKIT_BOND_CASHFLOW.

Note

Full list of available bond flow identifier : EXTENDED_FLOW_BOND_IDENTIFIER and others in ccmo.h file (FLOW_BOND_BALANCE ..)

```
*   ...
*       run_deal_ex(tid, pCmo);
*
*       double * bal_flow;
*       double * int_flow;
*       double * prin_flow;
*
*      //get_bond_flow_ex() returns a double pointer
*       bal_flow = get_bond_flow_ex(tid, "1A24",
     FLOW_BOND_BALANCE);
*       prin_flow = get_bond_flow_ex(tid, "1A24",
     FLOW_BOND_PRINCIPAL);
*       int_flow = get_bond_flow_ex(tid, "1A24",
     FLOW_BOND_INTEREST);
*
*       for (int i =0;i<MAX_PERIODS;i++)
*          //length of cashflow vector is MAX_PERIODS
*           printf("balance:%f,principal:%f,interest:%f \n",bal_flow[i],prin_flow[i],int_flow[i]);
*   ...
*
```

14.2 How to inspect underlying deals

14.2.1 Is this a reremic deal ?

Calling deal_has_underlying_deal() is a quick way to identify if a deal has underlying deals.

We can infer deal type by return value from deal_has_underlying_deal() :

return value	meaning
0	the deal only has normal loans as collateral
1	the deal only has tranches from underlying deals as collateral
2	the deal has a mix of normal loans and tranches from underlying deals as collateral

Sample code:

```
*   ...
*    open_deal_ex(tid, pCmo);
*    //quick way to identify deal type
*    printf("deal_has_underlying_deal: %d \n" ,deal_has_underlying_deal(tid));
*   ...
*
```

14.2.2 Inspect underlying deals

Let assume a dummy reremic deal structure as follows :

Deal Alpha(Parent deal)
|– Loan 1
|– Loan 2
|– Bond C from Deal Beta
Deal Beta(Underlying deal)
|– Loan 3
|– Loan 4
|– Loan 5
|– Bond D from Deal Lambda

Parent deal "Alpha" has 3 collaterals , one of them is a tranche "C" from deal "Beta".

Parent deal "Beta" has 4 collaterals , one of them is a tranche "D" from deal "Lambda".

Get underlying deal names

User can get underlying deal names from MARKIT_POOL_INFO::remic_deal_name.

Given dummy example above, code below will return output "id:3, name:Beta, full_name:Beta-C"

```
*   ...
*    open_deal_ex(tid, pCmo);
*    MARKIT_POOL_INFO* mpi = 0;
*    void* coll_it =obtain_collat_iterator(tid, 0);
*    while(mpi = get_next_collat(tid, coll_it)){
*            //MARKIT_POOL_INFO::remic_deal_name represents underlying deal name
*        printf("id:%d, name:%s ,full_name:%s \n ",mpi->loan_number,mpi->
      remic_deal_name,mpi->id);
*    }
*   ...
*
```

Get all underlying deals info

User can get details info about underlying deal by call view_reremic_deals(). This function will popluate underlying deal information to an array of CMO_STRUCT.

Given dummy example above, code below will list deal name "Beta","Lambda" .

Note

Calling this function requires all underlying deal files in the path set by set_input_path()

```
*  ...
*   open_deal_ex(tid, pCmo);
*   // get number of ALL underlying deals
*   int num_udy_deals = view_reremic_deals(tid, NULL, NULL);
*   printf("num of underlying deals: %d \n", num_udy_deals);
*   //allocate space for CMO_STRUCT array
*   CMO_STRUCT* udy_cmos = (CMO_STRUCT*)malloc(num_udy_deals*sizeof(
      CMO_STRUCT));
*   //popluate udy_cmos with CMO_STRUCTs.
*   view_reremic_deals(tid, NULL, udy_cmos);
*   //list all underlying deal id
*   for(int i=0;i<num_udy_deals;i++)
*       printf("underlying deal name: %s \n", udy_cmos[i].dealid);
*  ...
*
```

14.3 How to modify collateral using replace_collateral() ?

User can overide some data fields in MARKIT_POOL_INFO by calling replace_collateral().

Modify exsiting MARKIT_POOL_INFOs

First step is to use get all collateral structures MARKIT_POOL_INFOs, modified them ,and save address of each MARKIT_POOL_INFO.

Note

run_deal_ex() will reset modification done to the MAKRIT_POO_INFOs. Make sure replace_collateral() was called after before run_deal_ex().

```
*  ...
*   open_deal_ex(tid, pCmo);
*
*   // 2 dimensions pointer array to hold pointers to MARKIT_POOL_INFOs
*   MARKIT_POOL_INFO** mpi_list = (MARKIT_POOL_INFO**)malloc(sizeof(void*)
      * pCmo->num_colls);
*   // pointer points to the first element of mpi_list
*   MARKIT_POOL_INFO** p = mpi_list;
*   // mpi pointer to be popluated by get_next_collat()
*   MARKIT_POOL_INFO* mpi = 0;
*
*   void* coll_it =obtain_collat_iterator(tid, 0);
*   while(mpi = get_next_collat(tid, coll_it)){
*       printf("id:%d, balance:%f \n ",mpi->loan_number,mpi->
      current_balance);
*       // save address of mpi to mpi_list
*       *p = mpi;
*       // move pointer to next element in array mpi_list
*       p++;
*       if(mpi->loan_number > 10){
*           // modify attribute of MARKIT_POOL_INFO
*           mpi->current_balance *= 1.1;
*       }
*   }
*  ...
*
```

After overriding changes have been done, User should call replace_collateral() to replace current exsiting collaterals with updated ones.

```
*  ...
*   // replace collateral
*   replace_collateral(tid, NULL, mpi_list, pCmo->num_colls);
*  ...
*
```

Updated collateral can be retrived as following code:

```
*   ...
*    // inspect mpi after changes
*    coll_it =obtain_collat_iterator(tid, 0);
*    while(mpi = get_next_collat(tid, coll_it)){
*        printf("id:%d, balance:%f \n ",mpi->loan_number,mpi->
      current_balance);
*    }
*   ...
*
```

Chapter 15

Tutorial

Open deal This will show you how to open a deal.

View descriptive data This will show you how to view the descriptive data of deal, its bonds and collaterals.

Set assumption and projecting cashflows This will show you how to project cashflows base on assumptions.

15.1 Open deal

15.1.1 Open a CDOnet or SFW deal

Both CDOnet and SFW deal files ends with "SFW" extension, i.e "1776.SFW","BAM04003.SFW".

Procedure

- call set_input_path() to notify API where to look deal files
- create a null pointer and call create_deal_scenario_object() to initilize it
- create a CMO_STRUCT and assign a deal id(or a CUSIP/ISIN) to CMO_STRUCT::dealid, suggesting which deal to open
- call open_deal_ex()

Sample code

```
*  #include <stdio.h>
*  #include "wsa.h"
*
*  int main()
*  {
*   // Set the path where deals located, make sure there is "WSA_API.DB" database in this folder as well.
*   const char input_path[] = "C:\\Run.Bin";
*   // Tell API where to load deal files
*   set_input_path(input_path);
*   // initialize a identifier for open a deal
*   void* tid(NULL);
*   create_deal_scenario_object(&tid, NULL, NULL,false);
*   // create a CMO_STRUCT to hold deal information
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   // specify the deal name to a char array
*   char deal_id[] = "BAM04003";    // also can be a CUSIP/ISIN, e.g. "31364JMQ3"(CHS), "3128PS5H6"(AGENCY
      POOL), "000759BP4"(SFW), "36230RNB8"(HECM POOL), "00969QAJ0"(CDOnet), "US000780BW56"(SFW),
      "XS0333236890"(CDOnet)
*   // get length of deal name
*   int len = sizeof(pCmo->dealid) / sizeof(char);
*   // assign deal name to CMO_STRUCT::dealid
*   strncpy_s(pCmo->dealid,deal_id,len-1);
*   // open a deal , populate info to CMO_STRUCT and tid
*   open_deal_ex(tid, pCmo);
*   // after a deal opened, user can get which type deal it is.
*   ENGINE_TYPE current_engine = get_current_deal_engine(tid);
*   // 0 for CHS deal;1 stands for SFW deal;2 for CDOnet deal, please refer to enum type "ENGINE_TYPE"
*   printf("Current open deal with %d \n", current_engine);
*   // create a MARKIT_DEAL_INFO structure
*   MARKIT_DEAL_INFO deal_info;
*   // retrieve deal information and store them in MARKIT_DEAL_INFO instance
*   int ret = get_deal_info(tid, NULL, &deal_info);
*   // if ret is less than 0, user shall check the error message throw by get_deal_error_msg()
*   if(ret < 0)
*       printf("%s", get_deal_error_msg(tid));
*   else
*       //print deal information
*       printf("deal %s opened , it has %d bonds and %d collaterals \n", deal_info.
      deal_name, deal_info.num_bonds, deal_info.num_colls);
*   // close deal by calling close_deal_ex()
*   close_deal_ex(tid,pCmo);
*   // release tid
*   release_deal_scenario_object(&tid);
*   return 0;
*  }
*
```

15.1.1.1 Open a deal with settlement date

A SFW/CDOnet deal file may contain several deal updates. API will choose one of updates to open based on following rules:

if deal opened with CMO_STRUCT::settlement_date set to NULL (no settlement date):

- API will choose the latest deal update in the deal file.

if CMO_STRUCT::settlement_date is newer than any deal update in the deal file:

- API will choose the latest deal update available in the deal file to open.

if CMO_STRUCT::settlement_date is older than any deal update in the deal file:

- API will choose the oldest deal update available in the deal file to open.

if CMO_STRUCT::settlement_date is between the range of deal update in the deal file:

- API will choose the deal update whose update date is closest settlement date.

Note

If there are more than one updates in a month, select the latest one.

15.1.2 Open an agency deal.

Opening a agency deal is same way as opening a SFW/CDOnet deal.

Sample code

To open a CHS deal "02-073F.CHS", only need to use filename from ∗.CHS instead of ∗.SFW files.

```
*   ....
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*      char deal_id[] = "02-073F";
*      int len = sizeof(pCmo->dealid) / sizeof(char);
*      strncpy_s(pCmo->dealid,deal_id,len-1);
*   open_deal_ex(tid, pCmo);
*   ....
*
```

15.1.3 Open an agency pool.

Agency pool data resides in file "POOL_DATA_[yyyymm].LPD", opening an agency pool must specify settlement date. Make sure LPD file is in input path.

i.e ,if user set settlement date "7/14/13",then API will look for "POOL_DATA_201307.LPD" in input path.

Sample code

```
*   ....
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   // assign CUSIP to CMO_STRUCT::dealid
*   strncpy_s(pCmo->dealid,"3128HDWZ8", 10);
*   // set settlement date(mm/dd/yy) for opening, API will look file POOL_DATA_YYYYMM.LPD according to
       settlement date :
*     strncpy_s(pCmo->settlement_date, "05/01/14", 8);
*   open_deal_ex(tid, pCmo);
*   ....
*
```

Or:

```
*   ....
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   // assign "AGENCY" to CMO_STRUCT::dealid
*   strncpy_s(pCmo->dealid,"AGENCY", 6);
*   // assign "POOL" to CMO_STRUCT::bondid
*   strncpy_s(pCmo->bondid,"POOL", 4);
*   // assign CUSIP to CMO_STRUCT::bond.cusip
*   strncpy_s(pCmo->bond.cusip,"3128HDWZ8", 9);
*   // set settlement date(mm/dd/yy) for opening, API will look file POOL_DATA_YYYYMM.LPD according to
       settlement date :
*     strncpy_s(pCmo->settlement_date, "05/01/14", 8);
*   open_deal_ex(tid, pCmo);
*   ....
*
```

15.1.4 Open a HECM pool

HECM pools are in the datafile SFW_HECM_[YYYYMM].DB, opening a HECM pool must specify settlement date. Make sure DB file is in input path.

i.e ,if user set settlement date "7/14/13",then API will look for "SFW_HECM_201307.DB" in input path.

Sample code

```
*   ....
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   // assign CUSIP to CMO_STRUCT::dealid
*   strncpy_s(pCmo->dealid,"3620E1BG8", 10);
*   // set settlement date(mm/dd/yy) for opening, API will look file SFW_HECM_YYYYMM.DB according to
       settlement date :
*     strncpy_s(pCmo->settlement_date, "06/07/14", 8);
*   open_deal_ex(tid, pCmo);
*   ....
*  }
*
```

Or:

```
*   ....
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   // assign "SFW_HECM" to CMO_STRUCT::dealid
*   strncpy_s(pCmo->dealid,"SFW_HECM", 8);
*   // assign "POOL" to CMO_STRUCT::bondid
*   strncpy_s(pCmo->bondid,"POOL", 4);
*   // assign CUSIP to CMO_STRUCT::bond.cusip
*   strncpy_s(pCmo->bond.cusip,"3620E1BG8", 9);
*   // set settlement date(mm/dd/yy) for opening, API will look file SFW_HECM_YYYYMM.DB according to
       settlement date :
*     strncpy_s(pCmo->settlement_date, "06/07/14", 8);
*   open_deal_ex(tid, pCmo);
*   ....
*  }
*
```

15.1.5 Open with CUSIP

If user don't know which deal corresponding to a given CUSIP, API provide a function get_moodys_id() enable user to get deal/bond from a CUSIP.

Warning

Since one bond could have multiple CUSIPs, but in API we only popluate one of them to MARKIT_BOND_INFO.cusip or CCMO_BONDS_S.cusip. It is possible that user open a deal with CUSIP XXXXXXXXX, but the MARKIT_BOND_INFO.cusip and CCMO_BONDS_S.cusip are populated by CUSIP YYYYYYYY.

Sample code

```
*   ....
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   // assign CUSIP to CMO_STRUCT::dealid
*   strncpy_s(pCmo->dealid,"48124PAA2", 10);
*   // set settlement date(mm/dd/yy) for opening, API will look file SFW_HECM_YYYYMM.DB according to
      settlement date :
*     strncpy_s(pCmo->settlement_date, "06/07/14", 8);
*   open_deal_ex(tid, pCmo);
*   ....
*  }
*
```

Or:

```
*  #include <stdio.h>
*  #include "wsa.h"
*
*  int main()
*  {
*   const char input_path[] = "C:\\Run.Bin";
*   set_input_path(input_path);
*   void* tid(NULL);
*   create_deal_scenario_object(&tid, NULL, NULL,false);
*   // Given CUSIP:"48124PAA2"
*   const char cuisp[10] = "48124PAA2";
*   char deal_id[20]= "";
*   char bond_id[20]= "";
*   char error_message[30]= "";
*   // call get_moodys_id() to get corresponding deal id and bond id
*   // for pools ,deal_id will be either "AGENCY" or "SFW_HECM", and bond_id will be "POOL"
*   // for CDOnet/SFW deals, deal_id will be deal name and bond_id will be bond name.
*   if (get_moodys_id(cuisp,deal_id,sizeof(deal_id),bond_id,sizeof(bond_id),error_message,
      sizeof(error_message))==1){
*       printf("Retrieve DEAL/BOND successfully: deal:%s \n",deal_id);
*   }else{
*       printf("Fail to retrieve DEAL/BOND : message %s \n",error_message);
*   }
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*   // different open logic for "POOL" (SFW_HECM and Agency POOL)
*   if (!strncmp(bond_id,"POOL", sizeof(bond_id))){
*       strncpy_s(pCmo->bondid, bond_id, sizeof(bond_id));
*       strncpy_s(pCmo->bond.cusip,cuisp, sizeof(cuisp));
*   }
*     strncpy_s(pCmo->settlement_date, "06/07/14", 8);
*   open_deal_ex(tid, pCmo);
*   MARKIT_DEAL_INFO deal_info;
*   int ret = get_deal_info(tid, NULL, &deal_info);
*   if(ret < 0)
*       printf("%s", get_deal_error_msg(tid));
*   else
*       printf("deal %s opened , it has %d bonds and %d collaterals \n", deal_info.
      deal_name, deal_info.num_bonds, deal_info.num_colls);
*   close_deal_ex(tid,pCmo);
*   release_deal_scenario_object(&tid);
*   return 0;
*  }
*
*
```

15.2 View descriptive data

After deal was opened, user are able to retrieve descriptive information from deal file. API provide descriptive information in 3 levels: deal level, bond level and collateral level.

15.2.1 View deal information

There are three structures (CMO_STRUCT, MARKIT_DEAL_INFO and MOODYS_DEAL_INFO) contains the attributes of deal opened.

Deal update id

When a new update of SFW/CDOnet deal was published in deal library ,it will be assigned an unique number. For HECM and Agency pool, deal update id will return 0.

User can get the update id for SFW/CDOnet by calling get_deal_update_id() after open_deal_ex().

Sample code

```
*   #include <stdio.h>
*   #include "wsa.h"
*
*    int main()
*    {
*        const char input_path[] = "C:\\Run.Bin";
*        set_input_path(input_path);
*        void* tid(NULL);
*        create_deal_scenario_object(&tid, NULL, NULL,false);
*    char * deal_id = "1776";
*        CMO_STRUCT *pCmo = new CMO_STRUCT();
*        strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*        strncpy_s(pCmo->settlement_date, "06/07/14", 8);
*        open_deal_ex(tid, pCmo);
*
*    char update_id[10] = "";
*    // call get_deal_update_id to fill deal update id into variable "update_id"
*    get_deal_update_id(tid, update_id, sizeof(update_id));
*    printf("current deal update id is : %s \n",update_id);
*
*        close_deal_ex(tid,pCmo);
*        release_deal_scenario_object(&tid);
*        return 0;
*    }
*
```

CMO_STRUCT

CMO_STRUCT is populated after a deal was opened.

Sample code

```
*   #include <stdio.h>
*   #include "wsa.h"
*
*    int main()
*    {
*        const char input_path[] = "C:\\Run.Bin";
*        set_input_path(input_path);
*        void* tid(NULL);
*        create_deal_scenario_object(&tid, NULL, NULL,false);
*    char * deal_id = "1776";
*        CMO_STRUCT *pCmo = new CMO_STRUCT();
*        strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*       strncpy_s(pCmo->settlement_date, "06/07/14", 8);
*        open_deal_ex(tid, pCmo);
*    printf("Deal id from CMO_STRUCT: %s \n", pCmo->dealid);
*    printf("Next paydate from CMO_STRUCT: %s \n", pCmo->next_pay_date);
*        close_deal_ex(tid,pCmo);
*       release_deal_scenario_object(&tid);
*       return 0;
*   }
*
```

MARKIT_DEAL_INFO

MARKIT_DEAL_INFO can be populated by calling get_deal_info()

Sample code

```
*   ....
*     // view descriptive function must be called after deal opened
*     open_deal_ex(tid, pCmo);
*
*     // create a MARKIT_DEAL_INFO structure
*     MARKIT_DEAL_INFO deal_info;
*     // populate MARKIT_DEAL_INFO by calling get_deal_info(), to get parent deal info, pass NULL at second
        parameter.
*     int ret = get_deal_info(tid, NULL, &deal_info);
*     // use return value to see if calling is successful
*     if(ret < 0)
*         printf("%s", get_deal_error_msg(tid));
*     else
*         // if return value is greater than 0, then user can inspect MARKIT_DEAL_INFO for details
*         printf("deal %s opened , it has %d bonds and %d collaterals \n", deal_info.
      deal_name, deal_info.num_bonds, deal_info.num_colls);
*
*   ....
*
```

MOODYS_DEAL_INFO

MOODYS_DEAL_INFO contains additional deal information and can be populated by calling get_deal_info_ex().

Sample code

```
*   ....
*   void *pDeal = NULL;
*   //deal has been opened
*
*   MOODYS_DEAL_INFO mdi={};
*   int ret = get_deal_info_ex(pDeal, NULL, &mdi);
*   if (0 != ret)
*   {
*     // error handle
*   }
*
*   ....
*
```

15.2.2 View bond information

There are three structures contains bond information: "CCMO_BONDS_S", "MARKIT_BOND_INFO" and "MOODYS_BOND_INFO".

CCMO_BONDS_S

User are able to populate a single CCMO_BONDS_S by calling either get_bond_by_index_ex() or get_bond_by_name_ex().

Sample code

```
*
```

```
*   int main()
*   {
*       const char input_path[] = "C:\\Run.Bin";
*       set_input_path(input_path);
*       void* tid(NULL);
*       create_deal_scenario_object(&tid, NULL, NULL,false);
*   char * deal_id = "1776";
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*       open_deal_ex(tid, pCmo);
*
*   CCMO_BONDS_S Bond_info;
*   CCMO_BONDS_S* pBond_info = &Bond_info;
*   // populate CCMO_BONDS_S by index
*   printf("get_bond_by_index_ex() \n");
*   for (int i = 0;i< pCmo->num_bonds;i++){
*       get_bond_by_index_ex(tid, pBond_info,i);
*       printf("Bond:%s, Current Balance:%f \n",pBond_info->stripped_id,pBond_info->
      current_balance);
*   }
*   // get CCMO_BONDS_S by a bond name
*   get_bond_by_name_ex(tid, pBond_info,"A2");
*   printf("get_bond_by_name_ex() \n");
*   printf("Bond:%s, Current Balance:%f \n",pBond_info->stripped_id,pBond_info->
      current_balance);
*
*       close_deal_ex(tid,pCmo);
*       release_deal_scenario_object(&tid);
*       return 0;
*  }
*
*
```

User can populate all bonds info by calling view_all_bonds_ex() to avoid the looping.

Sample code

```
*  int main()
*  {
*   const char input_path[] = "C:\\Run.Bin";
*   set_input_path(input_path);
*   void* tid(NULL);
*   create_deal_scenario_object(&tid, NULL, NULL,false);
*   char * deal_id = "1776";
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*   open_deal_ex(tid, pCmo);
*
*   CCMO_BONDS_S bond_info_list[9] = {};
*   // populate CCMO_BONDS_S array by view_all_bonds_ex()
*   view_all_bonds_ex(tid, bond_info_list);
*   for(int i=0;i<sizeof(bond_info_list)/sizeof(CCMO_BONDS_S);i++){
*       printf("Bond: %s -> CUSIP: %s \n",bond_info_list[i].stripped_id, bond_info_list[i].cusip);
*   }
*
*   close_deal_ex(tid,pCmo);
*   release_deal_scenario_object(&tid);
*   return 0;
*  }
*
```

MARKIT_BOND_INFO

User can get bond information by call get_bond_info_by_index() with bond index or calling get_bond_info_by_-tranche() with bond name.

Sample code

```
*  int main()
*  {
*   const char input_path[] = "C:\\Run.Bin";
*   set_input_path(input_path);
*   void* tid(NULL);
*   create_deal_scenario_object(&tid, NULL, NULL,false);
*   char * deal_id = "1776";
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*   open_deal_ex(tid, pCmo);
*
*   MARKIT_BOND_INFO mbi = {};
*   // get MARKIT_BOND_INFO by index
*   for( int i =1;i<=pCmo->num_bonds;i++){
*       get_bond_info_by_index(tid, NULL, i, &mbi);
*       printf("Bond: %s pays %d per year \n",mbi.tranche_name,mbi.
      periodicity);
*   }
*   // get MARKIT_BOND_INFO by bond name
*   get_bond_info_by_tranche(tid, NULL, "A2", &mbi);
*   printf("coupon of Bond:A2 is %f \n",mbi.coupon);
*
*   close_deal_ex(tid,pCmo);
*   release_deal_scenario_object(&tid);
*   return 0;
*  }
*
```

MOODYS_BOND_INFO

MOODYS_BOND_INFO contains additional bond information and can be populated by calling get_bond_info_by_-index_ex() or get_bond_info_by_tranche_ex().

15.2.3 View collateral information

CCMO_POOL_INFO and CCMO_POOL_INFO_EX

User can use view_colls_ex() to populate all collaterals.

if user want to get all collateral by supplying -1 in 2nd parameter, make sure you have allocate the pool_info/pool_-info_ex size to number of collateral number

in this case ,pCmo->num_colls is 64.

sample code

```
*  int main()
*   {
*       const char input_path[] = "C:\\Run.Bin";
*       set_input_path(input_path);
*       void* tid(NULL);
*       create_deal_scenario_object(&tid, NULL, NULL,false);
*   char * deal_id = "1776";
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*       open_deal_ex(tid, pCmo);
*
*   // Create CCMO_POOL_INFO and CCMO_POOL_INFO_EX array to hold result.
*   CCMO_POOL_INFO pool_info[64]={};
*   CCMO_POOL_INFO_EX pool_info_ex[64]={};
*   // To populate all collateral info, call view_colls_ex() with -1 in second parameter
*   view_colls_ex(tid, -1, pool_info, pool_info_ex, sizeof(
      CCMO_POOL_INFO), sizeof(CCMO_POOL_INFO_EX), 0);
*   for(int i=0;i<64;i++){
*       printf("Collateral: %d, Balance: %f Periodicity: %d \n",i,pool_info[i].current_balance,pool_info_ex
      [i].periodicity);
*   }
*
*       close_deal_ex(tid,pCmo);
*       release_deal_scenario_object(&tid);
*       return 0;
*   }
*
```

MARKIT_POOL_INFO

MARKIT_POOL_INFO is also a structure providing collateral information. User can popluate this structure by calling get_next_collat().

Make sure call obtain_collat_iterator() to get an iterator for calling get_next_collat()

sample code

```
*  int main()
*  {
*   const char input_path[] = "C:\\Run.Bin";
*   set_input_path(input_path);
*   void* tid(NULL);
*   create_deal_scenario_object(&tid, NULL, NULL,false);
*   char * deal_id = "1776";
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*   open_deal_ex(tid, pCmo);
*
*   // create a MARKIT_POOL_INFO structure
*   MARKIT_POOL_INFO* mpi =0;
*   // getting iterator by calling obtain_collat_iterator()
*   void* coll_it =obtain_collat_iterator(tid, 0);
*   // loop over all available collateral under the deal opened
*   while (mpi = get_next_collat(tid, coll_it)){
*       printf("Loan ID: %d, Weight average coupon: %f \n", mpi->loan_number ,mpi->
     wac);
*   }
*
*   close_deal_ex(tid,pCmo);
*   release_deal_scenario_object(&tid);
*   return 0;
*  }
*
```

MOODYS_POOL_INFO

MOODYS_POOL_INFO contains additional collateral information. Users can populate this structure by calling get-_next_collat_ex().

Make sure to call obtain_collat_iterator_ex() to get an iterator for calling get_next_collat_ex().

sample code

```
*  void* tid = NULL;
*  CMO_STRUCT *pCmo = new CMO_STRUCT();
*  memset(pCmo, 0, sizeof(*pCmo));
*  strcpy(pCmo->dealid, "STATICLO");
*
*  set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*  assert(0 == open_deal_ex(tid, pCmo));
*
*  MOODYS_POOL_INFO* coll_info =0;
*  void* coll_it =obtain_collat_iterator_ex(tid, 0);
*  if(coll_it == 0)
*  {
*      std::cout << "Failure to start collat iteration " << get_deal_error_msg(tid) <<
     std::endl;
*  }
*  while(coll_info = get_next_collat_ex(tid,coll_it))
*  {
*      // do what you need with collateral
*  }
*
*  assert(0 == close_deal_ex(tid, pCmo));
*  delete pCmo;
*  pCmo = NULL;
*
```

15.3 Set assumption and projecting cashflows

15.3.1 Set rate assumption

Some collaterals and bonds generate interest base on reference rates. These reference rates are set to 0 by default. In order to apply these rates to reflect user's macro economy forecast, API provide set_rate_ex() function to set reference interest rate.

Get index of a bond

API provide reference rate in MARKIT_BOND_INFO::floater_index and CCMO_BOND_S::index. type of both fields are integer and maps to enumeration INDEX_TYPE, i.e, "1" stands for "LIBOR_1",Libor 1 month

sample code

```
*
*  int main()
*  {
*   const char input_path[] = "C:\\Run.Bin";
*   set_input_path(input_path);
*   void* tid(NULL);
*   create_deal_scenario_object(&tid, NULL, NULL,false);
*   char deal_id[9] = "AEG05005";
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*   printf("%d \n",open_deal_ex(tid, pCmo));
*
*   CCMO_BONDS_S *cbs = new CCMO_BONDS_S();
*   MARKIT_BOND_INFO *mbi = new MARKIT_BOND_INFO();
*
*   for (int i=1 ; i<=pCmo->num_bonds;i++){
*      //get_bond_by_index_ex(): bond index in third argument starts with 0
*       get_bond_by_index_ex(tid, cbs, i-1);
*      //get_bond_info_by_index(): bond index in third argument starts with 1
*       get_bond_info_by_index(tid, NULL, i, mbi);
*       printf("CCMO_BONS_S Bond:%s reference index is: %d \n", cbs->stripped_id, cbs->
     index);
*       printf("MARKIT_BOND_INFO Bond:%s reference index is: %d \n", mbi->
     tranche_name, mbi->floater_index);
*   }
*   close_deal_ex(tid,pCmo);
*   release_deal_scenario_object(&tid);
*   return 0;
*  }
*
*
```

Get indexes of a deal

API provide function get_required_rate_codes() to retrieve all rates used by current opened deal(Including index rate used by collateral and bond).

sample code

```
*
*   int main()
*   {
*       const char input_path[] = "C:\\Run.Bin";
*       set_input_path(input_path);
*       void* tid(NULL);
*       create_deal_scenario_object(&tid, NULL, NULL,false);
*   char deal_id[9] = "AEG05005";
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*       printf("%d \n",open_deal_ex(tid, pCmo));
*
*   // Initialize a empty array
*   int index_rate_used[MAX_INDEX_TYPES_EX] = {};
*   // Return integer indicates number of index used.
*   int num_index_rate = get_required_rate_codes(tid, index_rate_used,
     MAX_INDEX_TYPES_EX);
*   printf("There are total %d indexes used in this deal \n", num_index_rate);
*   // list index rate used by this deal
*   for(int i= 0 ;i<num_index_rate;i++){
*       printf("Index rate: %d \n",index_rate_used[i]);
*   }
*       close_deal_ex(tid,pCmo);
*       release_deal_scenario_object(&tid);
*       return 0;
*   }
*
```

Set a rate forecast

User are able to set a forecast interest rate via calling set_rate_ex(). User can set a flat rate for all projection periods or supplies with a vector to reflect a finer estimation.

flat rate assumption

```
*   ....
*    short libor_1_year = LIBOR_12;
*    double flat_libor_1_year = 0.012;
*    set_rate_ex(tid, &libor_1_year, 0, &flat_libor_1_year);
*   ....
*
```

vectorized rate assumption

To supply a vectorized rate assumption, make sure that the rate assumption starts with second element in the double array.

Warning

if cashflow size is different from the supplied rate vector, the last element of rate vector will repeat till the end of cashflow period .
i.e: cashflow size is 100 but only supplied with a 10-rates-array (0.01,0.02,0.03..0.1), the last element(0.1) will repeat till last projection period.

```
*   ....
*    short libor_1_month = LIBOR_1;
*    // IMPORTANT:the first element in array won't be used
*    double libor_1month[10] = {0,0.03,0.04,0.05,0.06,0.07,0.05,0.06,0.04,0.05};
*    set_rate_ex(tid, &libor_1_month, 10, libor_1month);
*   ....
*
```

sample code

```
*   int main()
*   {
*    const char input_path[] = "C:\\Run.Bin";
*    set_input_path(input_path);
*    void* tid(NULL);
*    create_deal_scenario_object(&tid, NULL, NULL,false);
*    char deal_id[9] = "AEG05005";
*    CMO_STRUCT *pCmo = new CMO_STRUCT();
*    strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*    printf("%d \n",open_deal_ex(tid, pCmo));
*    int index_rate_used[MAX_INDEX_TYPES_EX] = {};
*    int num_index_rate = get_required_rate_codes(tid, index_rate_used,
      MAX_INDEX_TYPES_EX);
*    printf("There are total %d indexes used in this deal \n", num_index_rate);
*
*    //array for rate forecast assumption
*    double libor_1month[10] = {0,0.03,0.04,0.05,0.06,0.07,0.05,0.06,0.04,0.05};
*    double libor_6month[10] = {0,0.04,0.05,0.06,0.07,0.08,0.06,0.07,0.05,0.06};
*
*    // set index rate forecast
*    for(int i= 0 ;i<num_index_rate;i++){
*        printf("Index rate: %d \n",index_rate_used[i]);
*        short index_rate_to_set = (short)index_rate_used[i];
*        // 1 is enum value for Libor 1 months in INDEX_TYPE
*        if (index_rate_to_set == 1)
*            set_rate_ex(tid, &index_rate_to_set, 10, libor_1month);
*        // 3 is enum value for Libor 6 months in INDEX_TYPE
*        if (index_rate_to_set == 3)
*            set_rate_ex(tid, &index_rate_to_set, 10, libor_6month);
*    }
*    run_deal_ex(tid, pCmo);
*
*    // Inspect the bond cashflow with rate assumption set
*    MARKIT_BOND_CASHFLOW* pMbc = get_bond_flow_ex1(tid, NULL, "IA3");
*    for(int j=0;j< pMbc->size;j++){
*        printf("Date:%d,Balance:%f,Rate:%f,Interest:%f \n", pMbc->dates[j],pMbc->
      balance[j],pMbc->rate[j], pMbc->interest[j]);
*    }
*
*    close_deal_ex(tid,pCmo);
*    release_deal_scenario_object(&tid);
*    return 0;
*   }
*
```

15.3.2 Set prepay assumption

User can set prepayment assumption on collaterals which will result in a prepayment in cashflow projections.

Make sure prepayment assumption is set before the deal run. set_prepayments_ex() is used to set prepayment assumption.

Prepay assumption on deal level

User can setup prepayment assumption on a deal level by supplying -1 to loan_num parameter.

In following sample code, API will run prepayment assumption on deal level at 0.013 annualized CPR.

```
*   ...
*    open_deal_ex(tid, pCmo);
*    //set prepayment assumption before run
*    double prepayment_assumption = 0.013;
*    //for deal level prepayment setting, use -1 to parameter "loan_num".
*    set_prepayments_ex(tid, PREPAY_CURVE_CPR, 0, &prepayment_assumption,
      -1, NULL);
*    run_deal_ex(tid, pCmo);
*   ...
*
```

Prepay assumption on loan level

User can setup prepayment assumption on a loan level by supplying ordinary loan index to parameter "loan_num".

This loan level setting will provide a finer prepay assumption when projecting collateral cashflow.

```
*   ...
*    open_deal_ex(tid, pCmo);
*    //set 3 different prepayment speed
*    double high_prepayment_assumption = 0.013;
*    double avg_prepayment_assumption = 0.07;
*    double low_prepayment_assumption = 0.02;
*
*    MARKIT_POOL_INFO* mpi =0;
*    void* coll_it =obtain_collat_iterator(tid, 0);
*    // apply 3 different prepayment speed to collateral level base on their FICO score
*    // in this code, it will assign a higher prepayment assumption to collateral with higher FICO score
*    while(mpi =  get_next_collat(tid,coll_it)){
*        if (mpi->fico >= 700)
*            set_prepayments_ex(tid, PREPAY_CURVE_CPR, 0, &
      high_prepayment_assumption, mpi->loan_number, NULL);
*        else if (mpi->fico >= 600)
*            set_prepayments_ex(tid, PREPAY_CURVE_CPR, 0, &
      avg_prepayment_assumption, mpi->loan_number, NULL);
*        else
*            set_prepayments_ex(tid, PREPAY_CURVE_CPR, 0, &
      low_prepayment_assumption, mpi->loan_number, NULL);
*    }
*    run_deal_ex(tid, pCmo);
*   ...
*
```

Vectorized prepay assumption

Similar to set_rate_ex(), user can set a prepayment vector other than a flat number for all periods.

Warning

Similar to set_rate_ex(), API won't use the first element of supplied array.

```
*  ...
*   open_deal_ex(tid, pCmo);
*   double prepayment_vector[10] = { 0, 0.01,0.02,0.03,0.04,0.05,0.07,0.06,0.05,0.04}
*   set_prepayments_ex(tid, PREPAY_CURVE_CPR, 10, prepayment_vector, -1,
      NULL);
*   run_deal_ex(tid, pCmo);
*  ...
*
```

sample code

```
*
* int main()
* {
*   const char input_path[] = "C:\\Run.Bin";
*   set_input_path(input_path);
*   void* tid(NULL);
*   create_deal_scenario_object(&tid, NULL, NULL,false);
*   char deal_id[9] = "AEG05005";
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   pCmo->actual_coll = 1;
*   strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*   // set prepayment after deal opened
*   open_deal_ex(tid, pCmo);
*   double prepayment_vector[10] = {0.01,0.02,0.03,0.04,0.05,0.07,0.06,0.05,0.04,0.03}
*   set_prepayments_ex(tid, PREPAY_CURVE_CPR, 10, prepayment_vector, -1,
      NULL);
*   run_deal_ex(tid, pCmo);
*   // inspect the prepayment from collateral cashflow
*   MARKIT_COLLAT_CASHFLOW *mcc = new
      MARKIT_COLLAT_CASHFLOW();
*   get_collateral_flow_ex1(tid, 0, NULL, mcc);
*   for(int i=0;i<mcc->size;i++)
*       printf("date:%d, balance:%f, prepayments:%f \n",mcc->dates[i], mcc->
      balance[i],mcc->prepayments[i]);
*
*   close_deal_ex(tid,pCmo);
*   release_deal_scenario_object(&tid);
*   return 0;
* }
*
*
```

15.3.3 Set default assumption

User can set default assumption on collaterals which will result in a default principals in cashflow projections.

Make sure default assumption is set before the deal run. set_defaults_ex() is used to set default assumption.

Default assumption on deal level

Default setting can apply to deal level.

```
*  ...
*   open_deal_ex(tid, pCmo);
*   //set default assumption before run
*   double default_assumption = 0.02;
*   //for deal level default setting, use -1 to parameter "loan_num".
*   set_defaults_ex(tid, DEFAULT_CURVE_CDR, 0, &default_assumption, -1,
      NULL);
*   run_deal_ex(tid, pCmo);
*  ...
*
```

Default assumption on loan level

Default setting can apply to loan/collateral level.

```
*  ...
*   open_deal_ex(tid, pCmo);
*   //set default assumption before run
*   double high_default_assumption = 0.05;
*   double avg_default_assumption = 0.025;
*   double low_default_assumption = 0.01;
*
*   MARKIT_POOL_INFO* mpi =0;
*   void* coll_it =obtain_collat_iterator(tid, 0);
*   // we assume higher FICO score leads to a lower probability of default
*   while(mpi =  get_next_collat(tid,coll_it)){
*       if (mpi->fico >= 700)
*           set_defaults_ex(tid, DEFAULT_CURVE_CDR, 0, &
     low_default_assumption, mpi->loan_number, NULL);
*       else if (mpi->fico >= 600)
*           set_defaults_ex(tid, DEFAULT_CURVE_CDR, 0, &
     avg_default_assumption, mpi->loan_number, NULL);
*       else
*           set_defaults_ex(tid, DEFAULT_CURVE_CDR, 0, &
     high_default_assumption, mpi->loan_number, NULL);
*   }
*   run_deal_ex(tid, pCmo);
*  ...
*
```

Vectorized default assumption

User can apply a default rate vector on deal level or collateral level.

Warning

Similar to set_rate_ex(), API won't use the first element of supplied array.

```
* ...
*   open_deal_ex(tid, pCmo);
*   double default_vector[10] = {0,0.01,0.02,0.03,0.04,0.05,0.07,0.06,0.05,0.04};
*   set_defaults_ex(tid, DEFAULT_CURVE_CDR, 10, default_vector, -1, NULL);
*   run_deal_ex(tid, pCmo);
* ...
*
```

sample code

```
*  int main()
*  {
*       const char input_path[] = "C:\\Run.Bin";
*       set_input_path(input_path);
*       void* tid(NULL);
*       create_deal_scenario_object(&tid, NULL, NULL,false);
*   char deal_id[9] = "AEG05005";
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*   pCmo->actual_coll = 1;
*       strncpy_s(pCmo->dealid, deal_id, sizeof(deal_id));
*
*   open_deal_ex(tid, pCmo);
*   double default_vector[10] = {0, 0.01,0.02,0.03,0.04,0.05,0.07,0.06,0.05,0.03};
*   set_defaults_ex(tid, DEFAULT_CURVE_CDR, 10, default_vector, -1, NULL);
*   run_deal_ex(tid, pCmo);
*
*   MARKIT_COLLAT_CASHFLOW *mcc = new
     MARKIT_COLLAT_CASHFLOW();
*   get_collateral_flow_ex1(tid, 0, NULL, mcc);
*
*   for(int i=0;i<mcc->size;i++)
*       printf("date:%d, balance:%f, default:%f \n",mcc->dates[i], mcc->
     balance[i],mcc->defaults[i]);
*
*       close_deal_ex(tid,pCmo);
*       release_deal_scenario_object(&tid);
*       return 0;
*  }
*
```

Chapter 16

New Feature List

Global ASSET_SENIORITY
: Subject to change

Class BANKLOAN_CALL_ADJ_PARAM
: Subject to change

Global BANKLOAN_JUNIOR_SENIOR_TYPE
: Subject to change

Global BUY_PRICE_OVERRIDE_TYPE
: Subject to change

Global CALL_DATE_OVERRIDE_TYPE
: Subject to change

Global CALL_OPTION_TYPE
: Subject to change

Global CALL_PRICE_OVERRIDE_TYPE
: Subject to change

Class CDO_DATE_INFO
: Subject to change

Global CDO_HAIRCUT_TYPE
: Subject to change

Class CDO_TEST_FLOW
: Subject to change

Class CDO_TEST_INFO
: Subject to change

Class COUPON_INFO
: Subject to change

Class DISTRESSED_PROPERTY_RECOVERY
: Subject to change

Class DPD_ASSUMPTION
: Subject to change

Global DPD_DISTRIBUTION_TYPE
: Subject to change

Class DPD_RESULT
: Subject to change

Class DPD_SCENARIO

Subject to change

Global DRAW_RATE_TYPE

Subject to change

Global ESG_RATING_TERM

Subject to change

Global ESG_RATING_TYPE

Subject to change

Global FLOW_MISC_INDENTIFIER

Subject to change

Global get_available_borrower_benefits (void ∗tid, const char ∗reremic_deal_id_or_null, BORROWER_BENEFIT_ELIGIBILITY benefit_list[], int size)

Subject to change

Global get_balloon_extension_assumptions (void ∗tid, const char ∗reremic_deal_id_or_null, int ∗months, double ∗rates, int length, int ∗delay, long loan_num)

Subject to change

Global get_bond_info_by_index_ex (void ∗tid, const char ∗reremic_deal_id_or_null, int index, MOODYS_BOND_INFO ∗bond_info)

Subject to change

Global get_bond_info_by_tranche_ex (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, MOODYS_BOND_INFO ∗bond_info)

Subject to change

Global get_cdo_date_info (void ∗tid, const char ∗reremic_deal_id_or_null, CDO_DATE_INFO ∗date_info)

Subject to change

Global get_cdo_test_flow (void ∗tid, TEST_TYPE test_type, const char ∗test_name, CDO_TEST_FLOW ∗flow_test)

Subject to change

Global get_cdo_test_info (void ∗tid, short ∗test_size, CDO_TEST_INFO ∗test_info)

Subject to change

Global get_currencies (void ∗tid, char ∗currencies[])

Subject to change

Global get_deal_account_flow (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗account_name, MOODYS_ACCOUNT_CASHFLOW ∗cf)

Subject to change

Global get_deal_info_ex (void ∗tid, const char ∗reremic_deal_id_or_null, MOODYS_DEAL_INFO ∗deal_info)

Subject to change

Global get_exchange_rate (void ∗tid, const char ∗currency, double ∗pval)

Subject to change

Global get_global_currency (void ∗tid, char ∗currency_index)

Subject to change

Global get_global_reinvestment (void ∗tid, GLOBAL_REINVESTMENT_INFO ∗reinv_info, short pool_size, GLOBAL_REINVESTMENT_ASSET_INFO pool_info[])

Subject to change

Global get_haircut_flow (void ∗tid, CDO_HAIRCUT_TYPE haircut_type)

Subject to change

Global get_index_rate (void ∗tid, const char ∗currency, short ∗idx)

Subject to change

Global get_indices (void ∗tid, const char ∗currency, short ∗ps_rates)

Subject to change

Global get_loan_dates (void ∗tid, int loan_number)

Subject to change

Global get_loan_flow (void ∗tid, int loan_number, const char ∗reremic_deal_id_or_null, int flow_identifier)

Subject to change

Global get_loan_flow_ex (void ∗tid, int loan_number, MOODYS_LOAN_CASHFLOW ∗cf)

Subject to change

Global get_loan_flow_size (void ∗tid, int loan_number)

Subject to change

Global get_misc_flow (void ∗tid, int flow_identifier)

Subject to change

Global get_moodys_ssfa_calc (void ∗tid, const char ∗bondid, MOODYS_SSFA_CALC ∗ssfa_calc)

Subject to change

Global get_pa_model_type (void ∗tid, char ∗pa_model_type, int pa_avail_vector[], int ∗avail_vector_num)

Subject to change

Global get_pa_vector (void ∗tid, int group_number, PA_POOL_VECTOR_TYPE identifier)

Subject to change

Global get_required_index_codes (void ∗tid, const char ∗currency, int ∗rate_codes, int size_of_array_-codes)

Subject to change

Class GLOBAL_REINVESTMENT_ASSET_INFO

Subject to change

Class GLOBAL_REINVESTMENT_INFO

Subject to change

Global INSURANCE_CLAIM

Subject to change

Global INT_CAPITAL_CODE_OVERRIDE

Subject to change

Global LIQUIDATION_PERIODICITY_TYPE

Subject to change

Class LOAN_PRICING_INPUT

Subject to change

Global LOAN_STATUS

Subject to change

Global MISSING_EXCHANGE_RATES_HANDLING

Subject to change

Class MONTE_CARLO_ASSUMPTION

Subject to change

Global MONTE_CARLO_CORRELATION_TYPE

Subject to change

Class MONTE_CARLO_DEF_PPY_REC_ASSUMPTION

Subject to change

Global MONTE_CARLO_DEFAULT_TYPE

Subject to change

Global MONTE_CARLO_OPTIMIZATION
  Subject to change

Class MONTE_CARLO_RESULT
  Subject to change

Class MOODYS_BOND_INFO
  Subject to change

Class MOODYS_DEAL_INFO
  Subject to change

Global MOODYS_RATING_TYPE
  Subject to change

Class MOODYS_SSFA_CALC
  Subject to change

Global MPA_ANALYSIS_PARAM
  Subject to change

Global MPA_ANALYSIS_PARAM_OFFSET
  Subject to change

Global MPA_ANALYSIS_TYPE
  Subject to change

Global MPA_MULTIPLIER_TYPE
  Subject to change

Global NON_PERFORMING_STATUS
  Subject to change

Global PA_ANALYSIS_TYPE
  Subject to change

Global PA_MULTIPLIER_TYPE
  Subject to change

Global PA_POOL_VECTOR_TYPE
  Subject to change

Global PAYMENT_FREQUENCY
  Subject to change

Global REINV_OVERRIDE_TYPE
  Subject to change

Global REINV_TERM_SETTING_TYPE
  Subject to change

Global REINV_TYPE
  Subject to change

Global replace_pa_pool_data (void ∗tid, int poolID, const char ∗paraName, const char ∗value)
  Subject to change

Global RESEC_EXCEPTIONS_HANDLING
  Subject to change

Global SERVICER_ADVANCES_BASE
  Subject to change

Global set_balloon_extension_assumptions (void ∗tid, const char ∗reremic_deal_id_or_null, int ∗months, double ∗rates, int length, int delay, long loan_num)
  Subject to change

Global set_borrower_benefits_rate (void ∗tid, const char ∗reremic_deal_id_or_null, short index, short vector, double ∗pval)

Subject to change

Global set_buy_price_override (void ∗tid, short override_type, double ∗price, int size)

Subject to change

Global set_call_date_override (void ∗tid, short override_type, char ∗override_date)

Subject to change

Global set_call_option (void ∗tid, short type, BOOLYAN set_sup_remic)

Subject to change

Global set_call_price_override (void ∗tid, short override_type, double ∗price, int size)

Subject to change

Global set_default_before_amortization (void ∗tid, BOOLYAN def_bef_amort, BOOLYAN set_sup_remic)

Subject to change

Global set_distressed_property_recovery (void ∗tid, int loan_number, DISTRESSED_PROPERTY_RECOVERY ∗recovery_inputs)

Subject to change

Global set_draw_rates (void ∗tid, short type, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)

Subject to change

Global set_exchange_rate (void ∗tid, const char ∗currency, double val)

Subject to change

Global set_global_reinvestment (void ∗tid, GLOBAL_REINVESTMENT_INFO reinv_info, short pool_size, const GLOBAL_REINVESTMENT_ASSET_INFO ∗pool_info)

Subject to change

Global set_index_rate (void ∗tid, const char ∗currency, short ∗idx, short vector, double ∗pval)

Subject to change

Global set_insurance_coverage (void ∗tid, const char ∗issuer, INSURANCE_CLAIM type, short is_vector, double ∗pval)

Subject to change

Global set_missing_exchange_rates_handling (MISSING_EXCHANGE_RATES_HANDLING handling)

Subject to change

Global set_mpa_confidence_level (void ∗tid, double confidence_level)

Subject to change

Global set_mpa_delinquent_pd (void ∗tid, double deq_30days, double deq_60days)

Subject to change

Global set_mpa_offset (void ∗tid, MPA_ANALYSIS_PARAM_OFFSET type, int unit, double offset)

Subject to change

Global set_mpa_optimization (void ∗tid, BOOLYAN toggle, double tail_percent, double opt_percent)

Subject to change

Global set_mpa_stress_range (void ∗tid, MPA_ANALYSIS_PARAM param_type, double floor, double cap)

Subject to change

Global set_pv_reinvest_override (void ∗tid, const char ∗bondid, short override_type)

Subject to change

Global set_reinvestment_type (void ∗tid, short reinv_type)

Subject to change

Global set_resec_exceptions_handling (RESEC_EXCEPTIONS_HANDLING handling)
Subject to change

Global set_whole_loan (void ∗tid, const WHOLE_LOAN_STRUCT ∗whole_loan, int length, int initial_date)
Subject to change

Global SIMULATION_TYPE
Subject to change

Global TEST_TYPE
Subject to change

Global UK_REGION
Subject to change

Global US_STATE
Subject to change

Global WHOLE_LOAN_COUPON_TYPE
Subject to change

Global WHOLE_LOAN_DEFAULT_METHOD_TYPE
Subject to change

Global WHOLE_LOAN_ISSUER_TYPE
Subject to change

Class WHOLE_LOAN_STRUCT
Subject to change

Global WHOLE_LOAN_TYPE
Subject to change

Chapter 17

Deprecated List

Global get_deal_surv_data (void ∗tid, MARKIT_DEAL_SURVEILLANCE_DATA ∗survData, int YYYYMM)
This method is deprecated.

Global get_markit_id (const char ∗id, char ∗deal, char ∗bond)
This method is deprecated, use get_moodys_id() instead.

Global get_markit_id1 (const char ∗id, char ∗deal, char ∗bond, char ∗err_buffer, int err_length)
This method is deprecated, use get_moodys_id() instead.

Global get_rates_ex (void ∗tid, short ∗ps_rates)
This method is deprecated. Use get_required_rate_codes().

Class MARKIT_DEAL_SURVEILLANCE_DATA
This structure is deprecated.

Global MARKIT_DEAL_SURVEILLANCE_DATA::groupLevelData [MAX_COLL_GROUPS]
This field is deprecated..

Class MARKIT_GROUP_SURVEILLANCE_DATA
This structure is deprecated.

Global OTHER
Same as POOL_TYPE_OTHER

Global set_deal_search_mode (DEAL_SEARCH_MODE mode)
This method is deprecated and does *NOT* affect deal search.

Global set_maximum_deal_scenario_objects (int max)
This method is deprecated and does *NOT* perform anything when it is called.

Chapter 18

Availability List

Global adjust_PA_vectors (void ∗tid, bool enable)
SFW

Global calc_cashflow_offset_ex (void ∗tid, const char ∗bondid, const char ∗settlement_date, int ∗months_-offset, int ∗days_accrued, int ∗days_offset)
CDOnet, CHS, SFW

Global calculate_bond_first_loss (void ∗tid, const char ∗bondid, FIRST_LOSS_INPUT first_loss_input, FIR-ST_LOSS_RESULT ∗first_loss_result)
SFW, CDOnet

Global clean_up_call_ex (void ∗tid, short state, long loan_num, BOOLYAN set_sup_remic)
CDOnet, CHS, SFW

Global clear_moodys_credit_model_setttings (void ∗tid)
SFW, CHS

Global close_deal_ex (void ∗tid, CMO_STRUCT ∗cmos)
ALL

Global create_deal_scenario_object (void ∗∗Tid, short ∗LogAction, char ∗LogFile, BOOLYAN ∗Debug)
ALL

Global dayt_to_str (DAYT julian, char ∗temp)
ALL

Global dayt_to_str_with_day_count (DAYT julian, char ∗temp, const int dayCount)
ALL

Global deal_has_underlying_deal (void ∗tid)
CDOnet, CHS, SFW

Global enable_bond_insurance (void ∗tid, const char ∗bondid, BOOLYAN is_enabled)
SFW

Global enable_default_on_snapshot_date (void ∗tid, BOOLYAN flag_snapshot)
SFW

Global enable_periodic_coupon_rate_projection (void ∗tid, BOOLYAN flag_periodic_rate)
SFW

Global enable_reinv_loan (void ∗tid, BOOLYAN populate_reinv_loan)
CDONET

Global enable_same_deal_multithreading (int flag)
ALL

Global enable_sfw_delinq_projection (void ∗tid, BOOLYAN is_enabled)

SFW

Global generate_cmm_custom_result_output (void ∗tid, char ∗custom_scen_name)

SFW

Global generate_forward_interest_rates (void ∗tid)

SFW, CDOnet, CHS

Global get_agency_pool_prefix (void ∗tid, char ∗pool_prefix)

CHS

Global get_asset_type_list (char ∗asset_type_list[], char ∗err_buffer, int err_length)

SFW, CHS, CDONET

Global get_available_borrower_benefits (void ∗tid, const char ∗reremic_deal_id_or_null, BORROWER_BE-NEFIT_ELIGIBILITY benefit_list[], int size)

SFW

Global get_average_collat (void ∗tid, void ∗collat_iterator, int group_number)

CDOnet, CHS, SFW

Global get_average_collat_by_bond (void ∗tid, void ∗collat_iterator, const char ∗bondid)

CHS, SFW

Global get_average_collat_for_managed_code (void ∗tid, void ∗collat_iterator, int group_number, MARKIT-_POOL_INFO ∗usr_pool, CCMO_ARM_INFO ∗arm, MARKIT_PAYMENT_SCHEDULE ∗sched, MARKIT_-PREPAY_PENALTY prepayPenalty[], int sizeOfPpenArray, int ∗hasArm, int ∗hasSched, int ∗hasPpen)

CDOnet, CHS, SFW

Global get_balloon_extension_assumptions (void ∗tid, const char ∗reremic_deal_id_or_null, int ∗months, double ∗rates, int length, int ∗delay, long loan_num)

SFW

Global get_bond_authorized_integral_amount (void ∗tid, char ∗bondid, double ∗value)

CDOnet

Global get_bond_band (void ∗tid, const char ∗bondid, double ∗pricing_wal, double ∗low, double ∗high)

CDOnet, CHS, SFW

Global get_bond_by_index_ex (void ∗tid, CCMO_BONDS_S ∗b, long index)

CDOnet, CHS, SFW

Global get_bond_by_name_ex (void ∗tid, CCMO_BONDS_S ∗b, const char ∗id)

CDOnet, CHS, SFW

Global get_bond_cf_date (int per, char ∗date, void ∗tid, const char ∗bondid)

CHS, SFW

Global get_bond_cf_dates (void ∗tid, const char ∗bondid)

CDOnet, SFW, CHS

Global get_bond_cf_length (void ∗tid, short path, const char ∗bondid)

CDOnet, SFW, CHS

Global get_bond_currency (void ∗tid, const char ∗bondid, char ∗currency)

All

Global get_bond_day_cal_cur_ex (void ∗tid, const char ∗bondid, BOOLYAN use_code, char ∗day_count, char ∗bus_rules, char ∗currency)

CDOnet, CHS, SFW

Global get_bond_FFIEC_results (void ∗tid, const char ∗bondid, FFIEC_INPUT_PARAMS ∗FFIEC_inputs, F-FIEC_RESULTS FFIEC_results[])

SFW

Global get_bond_flow_ex (void ∗tid, const char ∗bondid, int flow_identifier)

CDOnet, CHS, SFW

Global get_bond_flow_ex1 (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid)

CDOnet, CHS, SFW

Global get_bond_flow_ex1_for_managed_code (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE ∗cf)

CDOnet, CHS, SFW

Global get_bond_flow_sim (void ∗tid, short path, const char ∗bondid, int flow_identifier)

CDOnet, SFW

Global get_bond_implied_loss (void ∗tid, const char ∗bondid, double ∗implied_loss)

SFW

Global get_bond_index_ex (void ∗tid, const char ∗id)

CDOnet, CHS, SFW

Global get_bond_info_by_index (void ∗tid, const char ∗reremic_deal_id_or_null, int index, MARKIT_BOND-_INFO ∗bond_info)

CDOnet, CHS, SFW

Global get_bond_info_by_index_ex (void ∗tid, const char ∗reremic_deal_id_or_null, int index, MOODYS_B-OND_INFO ∗bond_info)

CDOnet, SFW

Global get_bond_info_by_tranche (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, M-ARKIT_BOND_INFO ∗bond_info)

CDOnet, CHS, SFW

Global get_bond_info_by_tranche_ex (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, MOODYS_BOND_INFO ∗bond_info)

CDOnet, SFW

Global get_bond_market_risk_metrics (void ∗tid, const char ∗bondid, METRIC_INPUT_STRUCT ∗metric_-inputs, METRIC_RESULTS_STRUCT ∗metric_results)

ALL

Global get_bond_market_risk_metrics_ex (void ∗tid, char ∗bondid, METRIC_ANCHORS anchor_type, dou-ble anchor_value, APPLY_SPREAD_TYPE apply_to, METRIC_RESULTS_STRUCT_EX ∗results_ex)

ALL

Global get_bond_misc_ex (void ∗tid, const char ∗Bond, BOOLYAN ∗IsSeg, BOOLYAN ∗IsMACR, BOOLYAN ∗IsPO)

CDOnet, CHS, SFW

Global get_bond_next_reset_date (void ∗tid, const char ∗bondid, int ∗next_reset_date)

SFW, CDOnet

Global get_bond_payflag (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid)

SFW,CDOnet

Global get_bond_payment_group (void ∗tid, const char ∗bondid, char ∗group_names[])

CHS

Global get_bond_rate_determination_date (void ∗tid, const char ∗bondid, int ∗determination_date)

CHS, SFW

Global get_bond_rate_reset_dates (void ∗tid, const char ∗bondid)

SFW

Global get_bond_rating_by_tranche (void ∗tid, const char ∗bondid, RATING_AGENCY agency, char ∗rating)

CDOnet,SFW

Global get_bond_step_up_coupon (void ∗tid, const char ∗bondid, BOND_STEP_UP_COUPON all_set_up_-coupons[], int array_size, int ∗num_available)

SFW

Global get_bond_total_loss (void ∗tid, const char ∗bondid, double ∗total_loss)

CDOnet, SFW, CHS

Global get_calculation_method (void ∗tid, const char ∗reremic_deal_id_or_null)

SFW

Global get_cdo_date_info (void ∗tid, const char ∗reremic_deal_id_or_null, CDO_DATE_INFO ∗date_info)

CDOnet

Global get_cdo_test_flow (void ∗tid, TEST_TYPE test_type, const char ∗test_name, CDO_TEST_FLOW ∗flow_test)

CDOnet

Global get_cdo_test_info (void ∗tid, short ∗test_size, CDO_TEST_INFO ∗test_info)

CDOnet

Global get_cf_date (int per, char ∗date, void ∗tid)

CDOnet, CHS, SFW

Global get_china_bond_info_by_tranche (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, CHINA_BOND_INFO ∗bond_info)

SFW

Global get_cleanup_call_ex (void ∗tid, char ∗CallDate, double ∗CallPct, int ∗CallPctCalc)

CDOnet, CHS, SFW

Global get_coll_cf_dates (void ∗tid)

CDOnet, SFW

Global get_coll_cf_length (void ∗tid, short path)

CDOnet, SFW

Global get_collateral_flow_ex (void ∗tid, long group_number, int flow_identifier)

CDOnet, CHS, SFW

Global get_collateral_flow_ex1 (void ∗tid, int group_number, const char ∗reremic_deal_id_or_null, MARKI-T_COLLAT_CASHFLOW ∗cf)

CDOnet, CHS, SFW

Global get_collateral_flow_sim (void ∗tid, short path, int flow_identifier)

CDOnet, SFW

Global get_collateral_id_ex (void ∗tid, const char ∗reremic_deal_id_or_null, int loan_index, const char ∗id-_type, char ∗id_array[], int id_array_length)

CHS

Global get_coupon_stepup_date (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗date)

SFW

Global get_currencies (void ∗tid, char ∗currencies[])

SFW, CDOnet, CHS

Global get_current_deal_engine (void ∗tid)

ALL

Global get_current_edf_scenario (void ∗tid)

CDOnet

Global get_current_moodys_cmm_scenario (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗cmm_-scenario)

SFW

Global get_current_mpa_scenario (void ∗tid)

SFW

Global get_current_pa_scenario (void ∗tid)

CHS, SFW

Global get_custom_call_status (void ∗tid, const char ∗reremic_deal_id_or_null, BOOLYAN ∗status)

SFW

Global get_dates_from_upd_ex (void ∗tid, char ∗szArchiveName, int UpdDate[])

CDOnet, CHS, SFW

Global get_deal_account_avail (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗account_names[], D-EAL_ACCOUNT_INFO account_info[], unsigned int account_size)

CDOnet, SFW

Global get_deal_account_flow (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗account_name, MO-ODYS_ACCOUNT_CASHFLOW ∗cf)

CDOnet, SFW

Global get_deal_calc_level (void ∗tid)

CDOnet, CHS, SFW

Global get_deal_error_msg (void ∗tid)

ALL

Global get_deal_fee (void ∗tid, const char ∗reremic_deal_id_or_null, MOODYS_FEE_STRUCT fee_info[], int size, int ∗num_fees)

SFW, CDOnet

Global get_deal_fee_flow (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗fee_name)

SFW, CDOnet

Global get_deal_hedge (void ∗tid, const char ∗reremic_deal_id_or_null, MOODYS_HEDGE_STRUCT hedge-_info[], int size, int ∗num_hedges)

SFW

Global get_deal_info (void ∗tid, const char ∗reremic_deal_id_or_null, MARKIT_DEAL_INFO ∗deal_info)

CDOnet, CHS, SFW

Global get_deal_info_ex (void ∗tid, const char ∗reremic_deal_id_or_null, MOODYS_DEAL_INFO ∗deal_info)

CDOnet, SFW, CHS

Global get_deal_issuer_type (void ∗tid, char ∗Issuer, char ∗Type)

CDOnet, CHS, SFW

Global get_deal_payment_group (void ∗tid, MARKIT_DEAL_PAYMENT_GROUP group_array[], int group_-array_size, int ∗num_available)

CHS

Global get_deal_refinance_date (void ∗tid, int refinance_dates_array[], int num_dates)

CDONET

Global get_deal_surv_data (void ∗tid, MARKIT_DEAL_SURVEILLANCE_DATA ∗survData, int YYYYMM)

CHS

Global get_deal_update_id (void ∗tid, char ∗const update_id, const int len)

CDOnet, CHS, SFW

Global get_dpd_current_default_timing (void ∗tid)

SFW

Global get_dpd_el_pd_factors (void ∗tid, double ∗el_factor, double ∗pd_factor)

SFW

Global get_dpd_results (void ∗tid, const char ∗bondid, DPD_RESULT ∗result)

SFW

Global get_dpd_revolving_default_timing (void ∗tid)

SFW

Global get_dpd_scenarios (void ∗tid, DPD_SCENARIO ∗scenarios, short size_scenarios)

SFW

Global get_dpd_threshold (void ∗tid, const char ∗rating, short year, double ∗threshold)

SFW

Global get_edf_scenarios (void ∗tid, char ∗scenario_list[])

CDOnet

Global get_exchange_rate (void ∗tid, const char ∗currency, double ∗pval)

SFW CDOnet

Global get_first_principal_pay_month (void ∗tid, const char ∗bondid, char ∗first_prin_pay_month)

CDOnet, SFW

Global get_forward_interest_rates (void ∗tid, const char ∗currency, short ∗rate_type)

SFW, CHS, CDONET

Global get_global_currency (void ∗tid, char ∗currency_index)

SFW, CDOnet, CHS

Global get_global_reinvestment (void ∗tid, GLOBAL_REINVESTMENT_INFO ∗reinv_info, short pool_size, GLOBAL_REINVESTMENT_ASSET_INFO pool_info[])

CDOnet

Global get_group_info (void ∗tid, const char ∗reremic_deal_id_or_null, int group_number, MARKIT_GROUP_INFO ∗group_info)

CDOnet, CHS, SFW

Global get_group_issue_date (void ∗tid, int group_number, char ∗date)

SFW

Global get_haircut_flow (void ∗tid, CDO_HAIRCUT_TYPE haircut_type)

CDOnet

Global get_hist_data_ex (void ∗tid, CMO_STRUCT ∗cmos, char ∗bondid, double hist_factor[], double hist_coupon[])

CDOnet, CHS, SFW

Global get_hist_data_ex1 (void ∗tid, CMO_STRUCT ∗cmos, char ∗bondid, int date[], double principal_losses[], double paid_interest[])

CHS, SFW

Global get_index_rate (void ∗tid, const char ∗currency, short ∗idx)

ALL

Global get_indices (void ∗tid, const char ∗currency, short ∗ps_rates)

ALL

Global get_input_path ()

ALL

Global get_last_principal_pay_month (void ∗tid, const char ∗bondid, char ∗last_prin_pay_month)

CDOnet, SFW

Global get_license_info (int num_features, LICENSE_INFO lic_info[])

SFW, CDOnet, CHS

Global get_loan_dates (void ∗tid, int loan_number)

SFW

Global get_loan_edf (void ∗tid, const char ∗reremic_deal_id_or_null, long loan_num, double pd[], int length)

CDOnet, SFW

Global get_loan_flow (void ∗tid, int loan_number, const char ∗reremic_deal_id_or_null, int flow_identifier)

SFW

Global get_loan_flow_ex (void ∗tid, int loan_number, MOODYS_LOAN_CASHFLOW ∗cf)

CDOnet, SFW

Global get_loan_flow_size (void ∗tid, int loan_number)

CDONET, SFW

Global get_loan_level_avail_YYYYMMs (void ∗tid, int YYYYMMs[], int sizeOfYYYYMMs, int ∗numAvailable)

CHS

Global get_loan_market_risk_metrics (void ∗tid, int LoanID, METRIC_INPUT_STRUCT ∗metric_inputs, METRIC_RESULTS_STRUCT ∗metric_results)

CDONet

Global get_loan_market_risk_metrics_ex (void ∗tid, int LoanID, METRIC_ANCHORS anchor_type, double anchor_value, APPLY_SPREAD_TYPE apply_to, METRIC_RESULTS_STRUCT_EX ∗results_ex)

ALL

Global get_loan_next_reset_date (void ∗tid, int loan_number, int ∗next_reset_date)

CDOnet

Global get_longest_ex (void ∗tid)

CDOnet, CHS, SFW

Global get_MA_rate_shifts_scenarios (const char ∗file_path, int yyyymmdd, char ∗scenario_list[])

CDOnet, SFW, CHS

Global get_markit_bond_pool_history (void ∗tid, const char ∗cusip, const int history_identifier, MARKIT_POOL_HISTORY_DATA pool_history[], int size_array, int YYYYMM)

CHS

Global get_markit_bond_pool_history_avail_YYYYMMs (void ∗tid, const char ∗cusip, int YYYYMMs[], int size_YYYYMMs, int ∗num_available)

CHS

Global get_markit_id (const char ∗id, char ∗deal, char ∗bond)

ALL

Global get_markit_id1 (const char ∗id, char ∗deal, char ∗bond, char ∗err_buffer, int err_length)

ALL

Global get_markit_pool_history (const char ∗cusip, const int history_identifier, MARKIT_POOL_HISTORY_DATA pool_history[], int size_array, int YYYYMM)

CHS

Global get_markit_pool_history_avail_YYYYMMs (const char ∗cusip, int YYYYMMs[], int size_YYYYMMs, int ∗num_available)

CHS

Global get_master_trigger_info (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗trigger_name, SBYTE ∗breached, char ∗sub_trigger_logic, char ∗sub_trigger_names[], char ∗sub_trigger_descs[], int size)

SFW

Global get_misc_flow (void ∗tid, int flow_identifier)

SFW CDOnet

Global get_monte_carlo_correlation (void ∗tid, MONTE_CARLO_CORRELATION_TYPE type, char ∗field1, char ∗field2, double ∗correlation)

CDOnet

Global get_monte_carlo_global_issuers (void ∗tid, char ∗issuer_names[], short size)

CDOnet

Global get_monte_carlo_result (void ∗tid, const char ∗bondid, MONTE_CARLO_RESULT ∗result)

CDOnet, SFW

Global get_moodys_bond_history (void ∗tid, const char ∗bondId, MOODYS_BOND_HISTORY bondHistory[], int sizeOfHistoryArray, int YYYYMM)

SFW CDOnet

Global get_moodys_bond_history_avail_YYYYMMs (void ∗tid, const char ∗bondId, int YYYYMMs[], int sizeOfYYYYMMs, int ∗numAvailable)

SFW CDOnet

Global get_moodys_cmm_scenarios (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗scenario_list[])

SFW

Global get_moodys_id (const char ∗id, char ∗deal, int deal_length, char ∗bond, int bond_length, char ∗err_buffer, int err_length)

ALL

Global get_moodys_pool_group_id (void ∗tid, const char ∗reremic_deal_id_or_null, int group_number, char ∗group_id)

SFW

Global get_moodys_pool_history (void ∗tid, int groupNumber, MOODYS_POOL_HISTORY poolHistory[], int sizeOfHistoryArray, int YYYYMM)

SFW

Global get_moodys_pool_history_avail_YYYYMMs (void ∗tid, int groupNumber, int YYYYMMs[], int sizeOfYYYYMMs, int ∗numAvailable)

SFW

Global get_moodys_ssfa_calc (void ∗tid, const char ∗bondid, MOODYS_SSFA_CALC ∗ssfa_calc)

SFW, CDOnet

Global get_mpa_economy_date (void ∗tid, int ∗year, int ∗quarter)

All

Global get_mpa_scenarios (void ∗tid, char ∗scenario_list[])

SFW

Global get_next_collat (void ∗tid, void ∗collat_iterator)

CDOnet, CHS, SFW

Global get_next_collat_ex (void ∗tid, void ∗collat_iterator)

CDOnet, SFW, CHS

Global get_next_collat_for_managed_code (void ∗tid, void ∗collat_iterator, MARKIT_POOL_INFO ∗usr_pool, CCMO_ARM_INFO ∗arm, MARKIT_PAYMENT_SCHEDULE ∗sched, MARKIT_PREPAY_PENALTY prepayPenalty[], int sizeOfPpenArray, int ∗hasArm, int ∗hasSched, int ∗hasPpen)

CDOnet, CHS, SFW

Global get_optional_redemption_date (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗date)

SFW

Global get_pa_default_pool_data (void ∗tid, const char ∗paraName, char ∗value, int &len)

CHS, SFW

Global get_pa_economy_date (void ∗tid, int ∗year, int ∗quarter)

All

Global get_pa_model_type (void ∗tid, char ∗pa_model_type, int pa_avail_vector[], int ∗avail_vector_num)

CHS,SFW

Global get_pa_scenarios (void ∗tid, char ∗scenario_list[])
CHS, SFW

Global get_pa_vector (void ∗tid, int group_number, PA_POOL_VECTOR_TYPE identifier)
CHS,SFW

Global get_pool_by_index_ex (void ∗tid, CCMO_POOL_INFO ∗p, long index)
CDOnet, CHS, SFW

Global get_pool_ptr_by_index_ex (void ∗tid, long index)
CDOnet, CHS, SFW

Global get_prospectus_prepayment_curves (void ∗tid, const char ∗reremic_deal_id_or_null, PPC_STRUCT all_PPCs[], int size, int ∗num_curves)
SFW

Global get_rate_ex (void ∗tid, short index)
CDOnet, CHS, SFW

Global get_rates_ex (void ∗tid, short ∗ps_rates)
CDOnet, CHS, SFW

Global get_reinv_recovery_rate (void ∗tid, long loan_num, double ∗recovery_rate)
CDOnet

Global get_reinv_weighted_avg_pd (void ∗tid, long loan_num, double pd[])
CDOnet

Global get_repline_index_list (void ∗tid, const char ∗reremic_deal_id_or_null, int loan_index, int repline_-array[], int repline_array_length)
CHS

Global get_required_index_codes (void ∗tid, const char ∗currency, int ∗rate_codes, int size_of_array_-codes)
ALL

Global get_required_rate_codes (void ∗tid, int ∗rate_codes, int size_of_array_codes)
CDOnet, CHS, SFW

Global get_reremic_bond_band (void ∗tid, char ∗dealid, const char ∗bondid, double ∗pricing_wal, double ∗low, double ∗high)
CHS, SFW

Global get_reremic_bond_misc (void ∗tid, char ∗dealid, const char ∗Bond, BOOLYAN ∗IsSeg, BOOLYAN ∗IsMACR, BOOLYAN ∗IsPO)
CDOnet, CHS, SFW

Global get_reremic_pool_ptr_by_index (void ∗tid, char ∗dealid, long index, int &error)
CHS, SFW

Global get_reremic_trigger_status (void ∗tid, char ∗dealid, char ∗trigger_name, SBYTE ∗status)
CHS, SFW

Global get_reremic_triggers_avail (void ∗tid, char ∗dealid, char ∗trigger_names[], char ∗trigger_descs[])
CHS, SFW

Global get_sdk_build_version ()
ALL

Global get_sub_trigger_info (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗sub_trigger_-name, char ∗sub_trigger_type, char ∗sub_trigger_operator, double ∗current_level, double ∗threshold, SBYTE ∗status, BOOLYAN ∗curable, SBYTE ∗override_type, int ∗override_date)
SFW

Global get_surv_avail_YYYYMMs (void ∗tid, int YYYYMMs[], int sizeOfYYYYMMs, int ∗numAvailable)
CHS

Global get_surveillance_data (void ∗tid, int YYYYMM, char ∗user_buffer, long size_of_user_buffer, long ∗actual_size)
CHS

Global get_trigger_avail_ex (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗trigger_names[], char ∗trigger_descs[], int ∗num_sub_triggers, int size)
SFW

Global get_trigger_status (void ∗tid, char ∗trigger_name, SBYTE ∗status)
CDOnet, CHS, SFW

Global get_triggers_avail (void ∗tid, char ∗trigger_names[], char ∗trigger_descs[])
CDOnet, CHS, SFW

Global get_trustee_loan_id (void ∗tid, const char ∗reremic_deal_id_or_null, int loan_number, char ∗trustee-_loan_id)
SFW

Global get_user_data_for_cb (void ∗tid)
CDOnet, CHS, SFW

Global ignore_asset_nonpayment_term (void ∗tid, bool val)
SFW

Global install_collat_assump_cb (void ∗tid, COLLAT_ASSUMP_CB collat_assump_cb)
CDOnet, CHS, SFW

Global install_collat_assump_cb_ex1 (void ∗tid, COLLAT_ASSUMP_CB_EX1 collat_assump_cb_ex1)
CDOnet, SFW, CHS

Global install_input_dir_callback (INPUT_DIR_CB callback)
CHS, SFW

Global install_input_dir_callback_ex (INPUT_DIR_CB_EX callback_ex)
CHS, SFW

Global install_per_period_assump_cb (void ∗tid, PER_PERIOD_ASSUMP_CB per_period_assump_cb)
CDOnet, CHS, SFW

Global install_pool_cashflow_cb (void ∗tid, POOL_CASHFLOW_CB pool_cashflow_cb)
CDOnet, CHS, SFW

Global install_user_cleanup_cb (void ∗tid, USER_CLEANUP_CB user_cleanup_cb, int invoke_on_deal_-close)
CDOnet, CHS, SFW

Global is_credit_sensitive_ex (void ∗tid, long loan_num)
CDOnet, CHS, SFW

Global load_MA_rate (const char ∗file_path, int yyyymmdd, const char ∗ma_scenario, const char ∗currency, short ∗idx)
CDOnet, SFW, CHS

Global load_MA_rates (void ∗tid, int yyyymmdd, const char ∗ma_scenario)
SFW, CHS, CDONET

Global load_MWSA_rate (const char ∗file_path, int yyyymmdd, const char ∗currency, short ∗idx)
CDOnet, SFW, CHS

Global load_MWSA_rates (void ∗tid, int yyyymmdd, BOOLYAN load_forward_curves)
SFW, CHS, CDONET

Global obtain_collat_iterator (void ∗tid, const char ∗reremic_deal_id_or_null)
CDOnet, CHS, SFW

Global obtain_collat_iterator_ex (void ∗tid, const char ∗reremic_deal_id_or_null)

CDOnet, SFW, CHS

Global open_deal_ex (void ∗tid, CMO_STRUCT ∗cmos)

ALL

Global open_pool_from_file (void ∗tid, const char ∗cusip, const char ∗reserved, int YYYYMMDD_-settlement_date)

ALL

Global price_bond (void ∗tid, const char ∗bondid, PRICING_ANCHORS anchorType, double anchorValue, PRICING_RESULTS ∗results)

CDOnet, CHS, SFW

Global price_loan (void ∗tid, int loan_number, PRICING_ANCHORS anchorType, double anchorValue, PRICING_RESULTS ∗results)

CDOnet, SFW

Global price_loan_ex (void ∗tid, int loan_number, LOAN_PRICING_INPUT pricing_param_input, PRICING_-RESULTS ∗results)

CDOnet, SFW

Global release_deal_scenario_object (void ∗∗Tid)

ALL

Global remove_simulation_cashflow_populated_limit (void ∗tid, BOOLYAN flag)

SFW

Global replace_collateral (void ∗tid, const char ∗reremic_deal_id_or_null, MARKIT_POOL_INFO ∗collat_-array[], int collat_array_size)

CDOnet, CHS, SFW

Global replace_collateral_for_managed_code (void ∗tid, const char ∗reremic_deal_id_or_null, MARKIT_P-OOL_INFO collat_array[], int collat_array_size)

CDOnet, CHS, SFW

Global replace_pa_pool_data (void ∗tid, int poolID, const char ∗paraName, const char ∗value)

CHS, SFW

Global run_deal_ex (void ∗tid, CMO_STRUCT ∗cmos)

ALL

Global run_default_probability_distribution (void ∗tid)

SFW

Global run_FFIEC_test (void ∗tid, int prepay_type, double ∗prepay_rates)

SFW

Global run_monte_carlo_simulation (void ∗tid)

CDOnet, SFW

Global set_addit_group_delinquencies (void ∗tid, const char ∗reremic_deal_id_or_null, int group_number, short is_vector, int delinq_type, double ∗dqVal)

CDOnet, CHS, SFW

Global set_balloon_extension_assumptions (void ∗tid, const char ∗reremic_deal_id_or_null, int ∗months, double ∗rates, int length, int delay, long loan_num)

SFW

Global set_bankloan_call_adj_param (void ∗tid, const BANKLOAN_CALL_ADJ_PARAM ∗bankloan_adj, int length)

CDONet

Global set_bond_cf_mode (void ∗tid, BOND_CF_MODE mode, BOND_PAYMENT_DATES_TYPE payment_-dates_type, int propagate_to_remics)

CDOnet, CHS, SFW

Global set_bond_flow (void ∗tid, const char ∗bondid, int flow_identifier, short flow_length, double ∗flows)

CDOnet, CHS, SFW

Global set_borrower_benefits_rate (void ∗tid, const char ∗reremic_deal_id_or_null, short index, short vector, double ∗pval)

SFW

Global set_buy_price_override (void ∗tid, short override_type, double ∗price, int size)

CDOnet

Global set_calculation_method (void ∗tid, PREPAY_DEFAULT_CALC_METHOD_TYPE method_index, BOOLYAN set_sup_remic)

SFW

Global set_call_date_override (void ∗tid, short override_type, char ∗override_date)

CDOnet

Global set_call_option (void ∗tid, short type, BOOLYAN set_sup_remic)

SFW

Global set_call_price_override (void ∗tid, short override_type, double ∗price, int size)

CDOnet

Global set_cdonet_dll_num (const int &num)

CDOnet

Global set_cdonet_unload_flag (bool unload_dll)

CDONET

Global set_cleanup_call (void ∗tid, double ∗percentage, CLEAN_UP_CALL_BALANCE_TYPE ∗call_balance_type, CLEAN_UP_CALL_LINK_TYPE ∗link_type, int ∗yyyymm_date, BOOLYAN set_sup_remic)

CHS, SFW

Global set_cmbs_loan_extension_assumption (void ∗tid, BOOLYAN use_default, BOOLYAN apply_flag, BOOLYAN non_perf_loan, int maturity_cutoff, int extend_years, double edf_threshold)

SFW

Global set_cmm_custom_scenario (void ∗tid, CMM_FACTOR_TYPE cmm_factor_type, CMM_FACTOR factor, const double ∗value, int length)

SFW

Global set_collateral_flow_ex (void ∗tid, long group_number, int flow_identifier, short flow_length, double ∗flows, CMO_STRUCT ∗cmo)

CDOnet, CHS, SFW

Global set_credit_card_assump_ex (void ∗tid, const char ∗reremic_deal_id_or_null, CREDIT_CARD_ASSUMP_TYPE assump_type, short is_vector, double ∗pval, long loan_num)

SFW

Global set_current_edf_scenario (void ∗tid, int idx)

CDOnet

Global set_current_moodys_cmm_scenario (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗cmm_scenario)

SFW

Global set_current_mpa_scenario (void ∗tid, int idx)

SFW

Global set_current_pa_scenario (void ∗tid, int idx)

SFW

Global set_custom_amortization_ex (void ∗tid, short newVal)

CDOnet, CHS, SFW

Global set_deal_account_default (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗account_-name, BOOLYAN account_default)

CDOnet, CHS, SFW

Global set_deal_calc_level (void ∗tid, CALC_LEVEL level, int propagate_to_remics)

CDOnet, CHS, SFW

Global set_deal_error_msg (void ∗tid, const char ∗err)

CHS, SFW

Global set_deal_fee_override (void ∗tid, const char ∗reremic_deal_id_or_null, int fee_id, short fee_type, double override_value)

SFW

Global set_deal_hedge_override (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗hedge_id, M-OODYS_HEDGE_OVERRIDE hedge_override_info)

SFW

Global set_deal_search_mode (DEAL_SEARCH_MODE mode)

ALL

Global set_default_before_amortization (void ∗tid, BOOLYAN def_bef_amort, BOOLYAN set_sup_remic)

CDOnet

Global set_default_from_ex (void ∗tid, const short type, BOOLYAN set_sup_remic)

SFW, CDOnet, CHS

Global set_default_non_performing_loans (void ∗tid, BOOLYAN is_defaulted, short ∗non_perf_status, BO-OLYAN set_sup_remic)

SFW CDOnet

Global set_default_till_end (void ∗tid, BOOLYAN val, BOOLYAN set_sup_remic)

SFW

Global set_defaults_ex (void ∗tid, short type, short is_vector, double ∗pval, long loan_num, BOOLYAN set-_sup_remic)

CDOnet, CHS, SFW

Global set_deferment_rates (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_-remic)

SFW

Global set_distressed_property_recovery (void ∗tid, int loan_number, DISTRESSED_PROPERTY_RECOV-ERY ∗recovery_inputs)

SFW

Global set_dpd_assumption (void ∗tid, const DPD_ASSUMPTION ∗assumption)

SFW

Global set_dpd_current_default_timing (void ∗tid, const double ∗timing, short size_timing, BOOLYAN sea-soning)

SFW

Global set_dpd_el_pd_factors (void ∗tid, double el_factor, double pd_factor)

SFW

Global set_dpd_revolving_default_timing (void ∗tid, const double ∗timing, short size_timing, BOOLYAN seasoning)

SFW

Global set_dpd_scenarios (void ∗tid, const DPD_SCENARIO ∗scenarios, short size_scenario)

SFW

Global set_dpd_threshold (void ∗tid, const char ∗rating, short year, double threshold)

SFW

Global set_draw_rates (void ∗tid, short type, short is_vector, double ∗pval, long loan_num, BOOLYAN set-_sup_remic)

SFW

Global set_edf_default_multiplier (void ∗tid, double multiplier)

CDOnet

Global set_engine_preference (const ENGINE_PREFERENCE &engine)

ALL

Global set_error_handling_level (ERROR_HANDLING_LEVEL level)

ALL

Global set_exchange_rate (void ∗tid, const char ∗currency, double val)

SFW CDOnet

Global set_forbearance_rates (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup-_remic)

SFW

Global set_FRA (void ∗tid, const char ∗currency, const char ∗rate_type, short start_month, short end_-month, double rate_value)

SFW, CDOnet, CHS

Global set_global_rates (const char ∗currency, short rate_size, short ∗rate_types, double ∗rate_values)

SFW, CDOnet, CHS

Global set_global_reinvestment (void ∗tid, GLOBAL_REINVESTMENT_INFO reinv_info, short pool_size, const GLOBAL_REINVESTMENT_ASSET_INFO ∗pool_info)

CDOnet

Global set_grace_rates (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)

SFW

Global set_index_rate (void ∗tid, const char ∗currency, short ∗idx, short vector, double ∗pval)

ALL

Global set_index_rate_ex (void ∗tid, const char ∗currency, short ∗idx, int num_paths, short rate_size, double ∗∗idx_val)

CDOnet, SFW, CHS

Global set_indiv_recovery_nonperf (void ∗tid, BOOLYAN use_indiv_recovery_nonperf)

CDONET

Global set_input_path (const char ∗input_path)

ALL

Global set_insurance_coverage (void ∗tid, const char ∗issuer, INSURANCE_CLAIM type, short is_vector, double ∗pval)

SFW

Global set_int_capital_code_override (void ∗tid, short int_capital_code_override_type)

SFW

Global set_liquidation_period (void ∗tid, const int period, long loan_num, BOOLYAN set_sup_remic)

SFW

Global set_liquidation_periodicity (void ∗tid, short liquidation_periodicity_type, BOOLYAN set_sup_remic)

SFW

Global set_liquidation_schedule (void ∗tid, short vector_length, double ∗pval, long loan_num, BOOLYAN set_sup_remic)

SFW

Global set_loan_edf (void ∗tid, const char ∗reremic_deal_id_or_null, long loan_num, double ∗pd, int length)

CDOnet, SFW

Global set_loan_lgd (void ∗tid, const char ∗reremic_deal_id_or_null, long loan_num, double ∗lgd, int length)

SFW

Global set_loan_schedule (void ∗tid, long loan_number, WHOLE_LOAN_SINK_FUND ∗sink_fund_info)

CDOnet

Global set_log_options (void ∗tid, short ∗LogAction, char ∗LogFile, BOOLYAN ∗Debug)

ALL

Global set_macroeconomic_factor_ex (void ∗tid, const char ∗country, short ∗factor_type, int num_paths, short val_size, double ∗∗factor_val)

SFW

Global set_metrics_input_ex (void ∗tid, METRIC_INPUT_STRUCT_EX ∗metric_inputs)

ALL

Global set_missing_exchange_rates_handling (MISSING_EXCHANGE_RATES_HANDLING handling)

ALL

Global set_missing_interest_rates_handling (MISSING_INTEREST_RATES_HANDLING handling)

ALL

Global set_monte_carlo_assumption (void ∗tid, const MONTE_CARLO_ASSUMPTION ∗basic_assumption, const MONTE_CARLO_DEF_PPY_REC_ASSUMPTION ∗def_ppy_rec_assumption)

CDOnet, SFW

Global set_monte_carlo_correlation (void ∗tid, MONTE_CARLO_CORRELATION_TYPE type, char ∗field1, char ∗field2, double correlation)

CDOnet

Global set_monte_carlo_default_time_and_recovery (void ∗tid, short num_path, short num_loan, short default_time, double recovery)

CDOnet,SFW

Global set_moodys_credit_model_settings (void ∗tid, const MOODYS_CREDIT_MODEL_SETTINGS credit-_model, BOOLYAN sets_up_only)

CDOnet, CHS, SFW

Global set_mpa_analysis_type (void ∗tid, MPA_ANALYSIS_TYPE type)

SFW

Global set_mpa_confidence_level (void ∗tid, double confidence_level)

SFW

Global set_mpa_custom_scenario (void ∗tid, const char ∗factor, const char ∗scope, const int ∗year, const int ∗quarter, const double ∗value, int length)

SFW

Global set_mpa_data_path (const char ∗path)

SFW

Global set_mpa_default_loan_data (void ∗tid, const char ∗loan_data_field, const char ∗value)

SFW

Global set_mpa_delinquent_pd (void ∗tid, double deq_30days, double deq_60days)

SFW

Global set_mpa_haircut (void ∗tid, short is_vector, double ∗pval, BOOLYAN seasoning)

SFW

Global set_mpa_insurance_non_payment (void ∗tid, double probability)

SFW

Global set_mpa_loan_cashflow (void ∗tid, BOOLYAN enable_loan_cf)

SFW

Global set_mpa_mid_course_adj (void ∗tid, BOOLYAN use)
SFW

Global set_mpa_multiplier (void ∗tid, MPA_MULTIPLIER_TYPE type, short is_vector, double ∗pval, long loan_num)
SFW

Global set_mpa_offset (void ∗tid, MPA_ANALYSIS_PARAM_OFFSET type, int unit, double offset)
SFW

Global set_mpa_optimization (void ∗tid, BOOLYAN toggle, double tail_percent, double opt_percent)
SFW

Global set_mpa_recovery_lag (void ∗tid, short is_vector, int ∗pval, long loan_num)
SFW

Global set_mpa_recovery_lag_by_state (void ∗tid, int judicial_lag, int non_judicial_lag)
SFW

Global set_mpa_simulation_length (void ∗tid, int length)
SFW

Global set_mpa_simulation_path_num (void ∗tid, int number)
SFW

Global set_mpa_stress_range (void ∗tid, MPA_ANALYSIS_PARAM param_type, double floor, double cap)
SFW

Global set_mpa_thread_count (void ∗tid, int number)
SFW

Global set_non_call_end (void ∗tid, int non_call_end_date)
CDOnet

Global set_non_perf_recovery_lag (void ∗tid, short value, BOOLYAN set_sup_remic)
SFW

Global set_pa_analysis_type (void ∗tid, PA_ANALYSIS_TYPE type)
CHS, SFW

Global set_pa_custom_scenario (void ∗tid, const char ∗factor, const int ∗year, const int ∗quarter, const double ∗value, int length)
CHS, SFW

Global set_pa_custom_scenario_ex (void ∗tid, const char ∗factor, const char ∗country, const char ∗region, const int ∗year, const int ∗quarter, const double ∗value, int length)
SFW,CHS

Global set_pa_data_path (const char ∗path)
CHS, SFW

Global set_pa_default_pool_data (void ∗tid, const char ∗paraName, const char ∗value)
CHS, SFW

Global set_pa_multiplier (void ∗tid, PA_MULTIPLIER_TYPE type, short is_vector, double ∗pval, long pool_-num)
SFW, CHS

Global set_pa_simulation_path_num (void ∗tid, int number)
SFW, CHS

Global set_pa_thread_count (void ∗tid, int number)
SFW, CHS

Global set_pool_level_user_data_for_cb (void ∗tid, const char ∗reremic_deal_id_or_null, short loan_num, void ∗user_data)
CHS, SFW

Global set_ppydef_only_on_paydate (void ∗tid, const char ∗reremic_deal_id_or_null, BOOLYAN only_on_-paydate)
SFW

Global set_prepay_default_compounding_method (void ∗tid, PREPAY_DEFAULT_COMPOUNDING_METH-OD prepay_default_compound)
SFW

Global set_prepayments_ex (void ∗tid, short type, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)
CDOnet, CHS, SFW

Global set_prospectus_prepayment_curves (void ∗tid, short PPC_index, int loan_num, BOOLYAN set_sup-_remic)
SFW

Global set_pv_reinvest_override (void ∗tid, const char ∗bondid, short override_type)
CDOnet

Global set_rate_ex (void ∗tid, short ∗idx, short vector, double ∗pval)
CDOnet, CHS, SFW

Global set_rate_shift_setting (void ∗tid, RATE_SHIFT_SETTING rate_shift)
CDOnet

Global set_realized_losses_at_liquidation (void ∗tid, BOOLYAN realized_at_liquidation, int months_prior_-liquidation, BOOLYAN set_sup_remic)
SFW

Global set_recover_at_maturity_call (void ∗tid, BOOLYAN is_enabled, BOOLYAN set_sup_remic)
SFW, CDOnet

Global set_recoveries_ex (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_-remic)
CDOnet, CHS, SFW

Global set_recovery_from (void ∗tid, short type, BOOLYAN set_sup_remic)
SFW

Global set_recovery_lag_ex (void ∗tid, short val, long loan_num, BOOLYAN set_sup_remic)
CDOnet, CHS, SFW

Global set_reinvestment_type (void ∗tid, short reinv_type)
CDOnet

Global set_reremic_default_from (void ∗tid, char ∗dealid, const short type)
CHS

Global set_reremic_defaults (void ∗tid, char ∗dealid, short type, short is_vector, double ∗pval, long loan_-num)
CHS, SFW

Global set_reremic_prepayments (void ∗tid, char ∗dealid, short type, short is_vector, double ∗pval, long loan_num)
CHS, SFW

Global set_reremic_recoveries (void ∗tid, char ∗dealid, short is_vector, double ∗pval, long loan_num)
CHS, SFW

Global set_reremic_recovery_lag (void ∗tid, char ∗dealid, short val, long loan_num)
CHS, SFW

Global set_reremic_service_advances (void ∗tid, char ∗dealid, const int type)
CHS, SFW

Global set_reremic_trigger_override (void ∗tid, char ∗dealid, char ∗trigger_name, short is_vector, SBYTE ∗override)

CHS, SFW

Global set_resec_exceptions_handling (RESEC_EXCEPTIONS_HANDLING handling)

CDOnet

Global set_resec_underlying_level (int level)

CDOnet

Global set_service_advances_ex (void ∗tid, const int type, BOOLYAN set_sup_remic)

CDOnet, CHS, SFW

Global set_service_advances_rates (void ∗tid, int group_number, short is_vector, double ∗pval)

SFW

Global set_service_advances_rates_type (void ∗tid, short type)

SFW

Global set_service_reimburse_advint (void ∗tid, const char ∗reremic_deal_id_or_null, BOOLYAN reimburse_advint)

SFW, CDOnet

Global set_sfw_dll_num (const int &num)

SFW

Global set_sfw_unload_flag (bool unload_dll)

SFW

Global set_simulation_engine (void ∗tid, short simulation_type)

CDOnet, SFW

Global set_smooth_losses (void ∗tid, BOOLYAN status, BOOLYAN set_sup_remic)

SFW

Global set_spot_spread (void ∗tid, const char ∗currency, ESG_RATING_TYPE rating_type, ESG_RATING_-TERM term_type, int num_paths, short rate_size, double ∗∗idx_val)

CDOnet

Global set_tmp_dir_treatment (TMP_DIR_TREATMENT tmp_dir_treatment)

ALL

Global set_trigger_override (void ∗tid, char ∗trigger_name, short is_vector, SBYTE ∗override)

CDOnet, CHS, SFW

Global set_trigger_override_ex (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗sub_trigger_-name, SBYTE override_type, int override_date)

SFW

Global set_up_ESG_model_interest_rates (ESG_MODEL_INPUTS ∗esg_inputs, ESG_CURRENCY_RATE_I-NPUTS esg_currency_inputs[], int esg_currency_inputs_size)

All

Global set_user_data_for_cb (void ∗tid, void ∗user_data)

CHS, SFW

Global set_whole_loan (void ∗tid, const WHOLE_LOAN_STRUCT ∗whole_loan, int length, int initial_date)

SFW, CDOnet

Global set_whole_loan_cumulative_rate (void ∗tid, double val, long loan_num)

SFW

Global set_whole_loan_default_method (void ∗tid, WHOLE_LOAN_DEFAULT_METHOD_TYPE type_index)

SFW

Global set_whole_loan_default_timing (void ∗tid, short vector_length, double ∗pval)

SFW

Global use_spot_values_for_initial_period (void ∗tid, bool enable)

SFW

Global view_all_bonds_ex (void ∗tid, CCMO_BONDS_S all_bonds[])

CDOnet, CHS, SFW

Global view_bond_coll_groups (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, int groups_array[], int groups_array_length, int ∗total_groups)

CHS, SFW

Global view_coll_groups (void ∗tid, int groups_array[], int groups_array_length, int ∗total_groups)

CDOnet, CHS, SFW

Global view_colls_ex (void ∗tid, short index, CCMO_POOL_INFO all_colls[], CCMO_POOL_INFO_EX all_colls_ex[], short pool_size, short pool_ex_size, short arm_size)

CDOnet, CHS, SFW

Global view_moodys_student_loan_info (void ∗tid, short index, MOODYS_STUDENT_LOAN_INFO all_colls[], short length)

SFW

Global view_reremic_colls_ex (void ∗tid, char ∗dealid, short index, CCMO_POOL_INFO all_colls[], CCMO_POOL_INFO_EX all_colls_ex[], short pool_size, short pool_ex_size, short arm_size)

CHS, SFW

Global view_reremic_deals (void ∗tid, char ∗dealid, CMO_STRUCT remics[])

CDOnet, CHS, SFW

Global write_log (void ∗tid, char ∗message, int log_level)

ALL

Chapter 19

Data Structure Index

19.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 20

File Index

20.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 21

Data Structure Documentation

21.1 BANKLOAN_CALL_ADJ_PARAM Struct Reference

bankloan information.

```
#include <wsa.h>
```

Data Fields

- MOODYS_RATING_TYPE moodys_rating

  Moodys Rating.
- ESG_RATING_TYPE esg_spotspread_rating

  The rating of the ESG spotspread path/trial, must be one of ESG_RATING_TYPE.
- double alpha

  Adjustment for adjusting ESG_spotspread curve of different ratings in bank loan call logic.
- double phi

  Adjustment for adjusting Discount Margin of loan of different ratings in bank loan call logic.

21.1.1 Detailed Description

New feature Subject to change

The documentation for this struct was generated from the following file:

- include/wsa.h

21.2 BANKLOAN_EXTEND_INFO Struct Reference

Data Fields

- short Moodys_Rating

  Moody's rating for this loan. should be one of the MOODYS_RATING_TYPE.
- short non_perfoming_flag

  0 or 1. By default is 0. 1 is checked
- double recovery_rate

  Individual recovery rate for the loan.
- double adjusted_spread

adjusted spread for the loan

- char loanx_id [20]

  loanx ID

- BANKLOAN_JUNIOR_SENIOR_TYPE seniority

  Seniority of the asset, refer to enum BANKLOAN_JUNIOR_SENIOR_TYPE.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.3 BOND_STEP_UP_COUPON Struct Reference

This structure is passed to get coupon step up date and trigger.

```
#include <wsa.h>
```

Data Fields

- char **trigger_name** [21]
- short date_flag

  This is trigger name.

- int set_up_date

  If TRUE, set_up_date is used, else set_up_date is not used.

- short reset_freq

  This is the date determine when the step executes.

- double coupon

  The number of months between interest rate resets.

- char int_type [20]

  This is coupon.

- double floater_spread

  This is description of the interest type.

- short index_type

  This is added to the index rate∗ multiplier.

- double floater_multiplier

  The market index used to calculate the coupon. Available values: INDEX_TYPE_EX and INDEX_TYPE.

- double swap_spread

  This is multiplied by the index rate to calculate the bond coupon at reset.

- short swap_index_type

  This is added to the swap index rate∗ multiplier.

- double swap_multiplier

  The swap index used to calculate the coupon. Available values: INDEX_TYPE_EX and INDEX_TYPE.

- double floater_cap

  This is multiplied by the swap index rate to calculate the bond coupon at reset.

- double floater_floor

  Floaters Only: The maximum interest rate for this bond.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.4 BORROWER_BENEFIT_ELIGIBILITY Struct Reference

This structure describes borrower benefit for a student loan.

```
#include <wsa.h>
```

Data Fields

- short benefit_type
- double rate

  Benefit rate.
- short month

  Number of months to be qualified for the benefit.
- BOOLYAN int_reamort

  Indicates if the interest benefit can be reamortized or not.
- short prin_cal_from

  Indicates which principal balance value should be taken for principal benefit calculation (0 for original balance; 1 for current balance), only available with benefit type=2(principal)

21.4.1 Detailed Description

See Also

MOODYS_STUDENT_LOAN_INFO

21.4.2 Field Documentation

21.4.2.1 short BORROWER_BENEFIT_ELIGIBILITY::benefit_type

Type of the borrower benefit:

Value	Meaning
0	Interest benefit with ACH transfer;
1	Interest benefit with on time payment
2	Principal benefit

The documentation for this struct was generated from the following file:

- include/wsa.h

21.5 CCMO_ARM_INFO Struct Reference

This structure holds adjustable rate information about collateral.

```
#include <ccmo.h>
```

Data Fields

- short index

  The market index. See INDEX_TYPE enum for types.
- BOOLYAN is_mega

  If true it is mega. If false it is not.
- short lookback

The number of days to look back to get the index rate for resets. Note: The rate period will be the prior month less the lookback.

- double multiplier

  The amount the index rate is multiplied by to calculate the rate the borrower pays.
- double gross_margin

  The amount added to the index rate to determine the rate the borrower pays.
- double net_margin

  The gross margin less fees (such as servicing). This is added to the index rate to determine the rate passed through to the deal.
- double lifetime_cap

  The maximum interest rate allowed. If this is an accumulation of collateral, it will be a weighted average.
- double lifetime_floor

  The minimum interest rate allowed. If this is an accumulation of collateral, it will be a weighted average.
- double periodic_cap

  The maximum absolute interest rate change per period. i.e., 2.0% means absolute rate change is at most 2.0%.
- double pay_periodic_cap

  The maximum percentage change in the payment per period. i.e., 0.075 = 7.5% maximum change.
- short first_reset

  Months to the first interest rate reset based on the date the deal is opened as of.
- short first_pay_reset

  Months to the first payment reset based on the date the deal is opened as of.
- short reset_freq

  The months between interest rate resets.
- short pay_reset_freq

  The months between payment resets.
- double neg_amort_limit

  The maximum lifetime percentage increase in balance due to negative amortization.
- double first_pay_cap

  The maximum absolute payment change for the first reset based on the date the deal is opened as of.
- double first_reset_cap

  The maximum absolute interest rate change for the first reset based on the date the deal is opened as of.
- short recast_month

  Reserved for future use.
- short original_first_reset

  Months to original first rate reset (important for hybrids)
- double original_coupon

  Coupon at origination.
- short original_first_pay_reset

  Months to original first payment reset.
- short consecutive_recast_months

  Months between requests after the first one.

21.5.1 Detailed Description

Some of the ARM information is duplicated in the CCMO_POOL_INFO for backward compatibility.

See Also

- CCMO_POOL_INFO
- CCMO_POOL_INFO_EX
- MARKIT_POOL_INFO
- view_colls_ex()
- view_reremic_colls_ex()

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.6 CCMO_BONDS_S Struct Reference

This structure holds individual bond information.

```
#include <ccmo.h>
```

Data Fields

- double orig_balance

  The original balance of the bond when the deal is issued.
- double current_balance

  The bond outstanding principal as of the date the deal is opened.
- BOOLYAN notional

  This field is true if the bond has a notional balance i.e. it is an interest only bond.
- double coupon

  The coupon as of the status date for which the deal is opened (see CMO_STRUCT.settlement_date).
- DAYT date
- short delay_days

  The number of days delay between end of interest accrual and bond payment.
- char id [20]

  The bond ID, including the deal ID.
- char stripped_id [20]

  The bond ID, stripped of the deal ID.
- char cusip [10]

  The CUSIP for the bond.
- BOOLYAN z_bond

  This field is true if the bond is an accrual bond.
- DAYT stated_maturity

  The bond's stated maturity expressed as a long integer. Use dayt_to_str() to convert to mm/dd/yy.
- char prin_type [20]

  A description of the principal type.
- char int_type [20]

  A description of the interest type.
- double cap

  Floaters Only: The maximum interest rate for this bond.
- double floor

  Floaters Only: The minimum interest rate for this bond.
- double spread

  Floaters Only: This is added to the current index rate ∗ multiplier.

- double multiplier

  Floaters Only: This is multiplied by the current index rate to calculate the bond coupon at reset.
- short index

  Floaters Only: The market index used to calculate the coupon.
- short index2

  Floaters Only: An optional second index used to calculate the coupon.
- short reset

  Floaters Only: Months between rate resets.
- double per_adj_cap

  Floaters Only: The maximum change in coupon for one reset.
- short lockout

  Floaters Only: The number of months before the coupon can be adjusted.
- short last_float_per

  Floaters Only: The last period the coupon can be adjusted.
- double resume_coupon

  Floaters Only: Coupon rate after the last_float_per.
- short component
- short type

21.6.1 Detailed Description

See Also

- CMO_STRUCT
- MARKIT_BOND_INFO
- get_bond_by_index_ex()
- get_bond_by_name_ex()
- view_all_bonds_ex()

21.6.2 Field Documentation

21.6.2.1 short CCMO_BONDS_S::component

Flag field indicating if bond is a component bond.

Value	Meaning
>0	this is a component.
<0	this is a owner.
0	normal bond

The absolute values of the owner bond and the components component field are the same.

21.6.2.2 DAYT CCMO_BONDS_S::date

This is the beginning of the bond accrual period for the current distribution date. This is in DAYT format. Please use dayt_to_str_with_day_count() to convert to mm/dd/yy. Parameter dayCount can be determined by MARKIT_BON-D_INFO::day_count.

21.6.2.3 short CCMO_BONDS_S::type

Parity Type

Value	Meaning
0	Internal use
4	Tranche principal payment type is SEQ
5	Z bond
6	Principal only bond
7	Interest only bond

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.7 CCMO_COLLAT_ASSUMPTIONS Struct Reference

This structure is passed to the per-collateral assumptions call back function. Populate its element with assumptions for all amortization periods. For use of the per collateral assumptions call back function, populate assumptions for a given collateral for the entire simulation period.

```
#include <ccmo.h>
```

Data Fields

- int prepay_type

  Unit of prepayment used. See set_prepayments_ex() function for available values.
- int n_prepays

  Length of prepay projection periods.
- double ∗ prepay_vector
- int default_type

  Unit of default used. See set_defaults_ex() function for available values.
- int n_defaults

  Length of default projection periods.
- double ∗ default_vector

  Vector of default values. See prepay_vector for more info.
- int n_recoveries

  Number of recovery rates provided.
- double ∗ recovery_vector

  Vector of recovery rates. See prepay_vector for more info.
- int n_recovery_lags

  Number of recovery lags provided.
- short ∗ recovery_lag_vec

  Vector of recovery lag values in months. See prepay_vector for more info.
- int n_delinquencies

  Number of delinquencies provided.
- double ∗ dlnq [GROUP_DELINQ_SIZE]

  Vector of 30 / 60 / 90+ days delinquencies in decimals.
- int servicer_advancing

  Flag indicating servicer if there is any advances from the services. See set_service_advances_ex().
- double ∗ draw_rates_vector

  Vector of annualized draw rates for HELOC in decimals. Element 1 corresponds to the next payment date.
- int n_draw_rates

  Number of values in draw_rates_vector.
- int additional_loss_unit_type

  Additional loss rate unit type. reference to ADDITIONAL_LOSS_UNIT.

- double ∗ additional_losses [ADDITIONAL_LOSS_TYPE_SIZE]

  Additional loss rates. This rates will be passed to CCMO_PERIOD_ASSUMPTIONS.monthly_additional_losses.
- int n_additional_losses

  The length of additional losses array.

21.7.1 Detailed Description

See Also

install_collat_assump_cb()

21.7.2 Field Documentation

21.7.2.1 double∗ CCMO_COLLAT_ASSUMPTIONS::prepay_vector

Vector of prepayment values. It should be populated starting from element 1. The size of the vector does not exceeds MAX_PERIODS. Numbers should be provided in decimals. The last value will be propagated to the end of simulation.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.8 CCMO_PERIOD_ASSUMPTIONS Struct Reference

This structure is passed to per-period assumption call back function. Populate the period's element of the arrays with assumptions for the current period. For use of the per period assumptions call back functions, populate the period's element of the arrays to provide assumptions for the current period. All arrays have a size of MAX_PERIODS.

```
#include <ccmo.h>
```

Data Fields

- double ∗ smm_vector

  Single month mortality vector expressed in decimals. 1 SMM would be 0.01.
- double ∗ mdr_vector

  Monthly default rate vector expressed in decimals. 1 MDR would be 0.01.
- double ∗ recovery_vector

  Monthly vector of recoveries expressed in decimals. 80% recovery would be 0.8.
- short ∗ recovery_lag_vec

  Vector of the lags between foreclosure and recovery expressed in months.
- double ∗ dlnq [GROUP_DELINQ_SIZE]

  Monthly vector of percentage of delinquent loans, expressed in decimals, for 30, 60, 90+ days delinquencies.
- int servicer_advancing

  Flag indicating if servicer advances principal or/and interest, See set_service_advances_ex() for available values.
- double ∗ monthly_draw_rates_vector

  Monthly vector of draw rates in decimals.
- double ∗ monthly_additional_losses [ADDITIONAL_LOSS_TYPE_SIZE]

  Monthly additional rates come from CCMO_COLLAT_ASSUMPTIONS.additional_losses.

21.8.1 Detailed Description

See Also

install_per_period_assump_cb()

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.9 CCMO_POOL_INFO Struct Reference

This structure holds information about collateral.

```
#include <ccmo.h>
```

Data Fields

- char id [20]

  Identifies the piece of collateral. Collateral id - "MBS" or deal name.
- long number

  For internal use. ONLY USED FOR READING FRED.MTG FILES.
- short type

  The type of collateral. Must be enum of POOL_TYPE. This corresponds to CCMO_POOL_INFO::type_str.
- char type_str [10]

  The description of the type.
- double pass_through

  The current pass_through rate as of the date the deal is opened. This will be net of trustee and servicing fees. If this is an accumulation of collateral, it will be a weighted average.
- double original_balance

  The original balance of the collateral. If this is an accumulation of collateral, it will only include collateral that is still outstanding.
- double factor

  The factor of the collateral. The current balance is the factor $*$ original_balance.
- double gross_coupon

  The highest gross coupon of the collateral in an agency pool. Irrelevant for non-agencies. FOR BOND VALUE COMPUTATION.
- double wac

  The actual gross coupon of the collateral piece. If this is an accumulation of collateral, it will be a weighted average.
- short term

  The longest original maturity (in months) of the collateral in an agency pool. Irrelevant for non-agencies. FOR BOND VALUE COMPUTATION.
- short wam

  The remaining months until maturity (as of the date the deal is opened). If this is an accumulation of collateral, it will be a weighted average.
- short original_term

  The original term in months. If this is an accumulation of collateral, it will be a weighted average.
- double psa_coupon

  Strip PO: The coupon used by prepayment models (the actual coupon is 0.) Irrelevant otherwise, set to the pass-through rate. FOR GENERIC PREPAY MODELING.
- short wala

  The age of the collateral (as of the date the deal is opened). If this is an accumulation of collateral, it will be a weighted average.

- short balloon_period

  The original balloon term.
- short coll_group

  The collateral group this piece belongs to. All collateral in a bucket will belong to the same collateral group. COLLATERAL GROUP.
- double level_pay

  The current level payment for the collateral. If 0, the payment will be calculated.
- double price

  Reserved for future use.
- short prepay_lockout

  The remaining prepayment lockout in months (as of the date the deal is opened).
- short yield_maintain

  Reserved for future use.
- short day_count

  Reserved for future use.
- short penalty_4

  Reserved for future use.
- short penalty_3

  Reserved for future use.
- short penalty_2

  Reserved for future use.
- short penalty_1

  Reserved for future use.
- short prin_lockout

  The remaining principal lockout in months (as of the date the deal is opened).
- short forward_purchase

  The remaining forward purchase months (as of the date the deal is opened).
- BOOLYAN io

  If true (non-zero), collateral is interest-only.
- short arm_index

  ARM only – the market index. See INDEX_TYPE enum for types.
- short first_reset

  ARM only – months to the next interest rate reset.
- short reset_freq

  ARM only – months between interest rate resets.
- short first_pay_reset

  ARM only – months to the next payment reset.
- short pay_reset_freq

  ARM only – months between payment resets.
- double net_margin

  ARM only – amount added to the index to calculate pass-through rate. If this is an accumulation of collateral, it will be a weighted average.
- double gross_margin

  ARM only – amount added to the index to calculate the gross interest rate for the collateral. If this is an accumulation of collateral, it will be a weighted average.
- double periodic_cap

  ARM only – The maximum absolute rate change allowed in one period. If this is an accumulation of collateral, it will be a weighted average.
- double lifetime_cap

  ARM only – The maximum interest rate allowed. If this is an accumulation of collateral, it will be a weighted average.
- double first_reset_cap

ARM only – The maximum absolute rate change allowed at the first rate reset. If this is an accumulation of collateral, it will be a weighted average.

- double life_cap_spread

  ARM only – The maximum amount the coupon can vary from the original coupon over the life of the collateral. If this is an accumulation of collateral, it will be a weighted average.

- double current_balance

  Current unpaid balance (as of the date the deal is opened).

- double ∗ bv_ratio

  Not currently used.

21.9.1 Detailed Description

See Also

- CMO_STRUCT
- CCMO_POOL_INFO_EX
- MARKIT_POOL_INFO
- get_pool_by_index_ex()
- get_pool_ptr_by_index_ex()
- get_reremic_pool_ptr_by_index()
- view_colls_ex()
- view_reremic_colls_ex()

The documentation for this struct was generated from the following file:

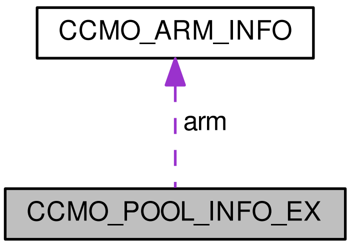
- include/ccmo.h

21.10 CCMO_POOL_INFO_EX Struct Reference

This structure holds information about collateral.

```
#include <ccmo.h>
```

Collaboration diagram for CCMO_POOL_INFO_EX:



Data Fields

- CCMO_ARM_INFO ∗ arm

  The structure containing adjustable rate information. This should either be allocated or set to NULL.
- short periodicity

  The number of payments per year.
- double avg_loan_bal

  The average loan balance.
- double ltv

  Loan To Value.
- short fico

  FICO score.
- char servicer_seller [11]

  The servicer/seller for the collateral.
- char delinquency [5]

  The delinquency status of the collateral.
- char state [3]

  The two-character state code.
- char country [4]

  The three-character country code.
- char purpose [5]
- char property_type [5]
- char occupancy [5]

21.10.1 Detailed Description

This structure is an extension to CCMO_POOL_INFO.

See Also

- CCMO_POOL_INFO
- MARKIT_POOL_INFO
- view_colls_ex()
- view_reremic_colls_ex()

21.10.2 Field Documentation

21.10.2.1 char CCMO_POOL_INFO_EX::occupancy[5]

The type of occupancy.

Value	Meaning
1	Primary
2	Secondary
3	Investment
4	Unknown

21.10.2.2 char CCMO_POOL_INFO_EX::property_type[5]

The type of property.

Value	Meaning
0	Other
1	Single Family
2	Multi-Family
3	Condo
4	PUD
5	Commercial
6	Coop
7	Mobile Home
8	Manufactured Housing
9	Not Available
10	Duplex
11	Triplex
12	Fourplex
13	5+ Units

21.10.2.3 char CCMO_POOL_INFO_EX::purpose[5]

The purpose of the loan.

Value	Meaning
0	Other
1	Purchase
2	Cash Out Refinance
3	Home Improvement
4	New Construction
5	Rate Term Refinance
6	Not Available

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.11 CDO_DATE_INFO Struct Reference

This struct stores the date info of the CDOnet Deal.

```
#include <wsa.h>
```

Data Fields

- int closing_date

  Deal closing date, format "YYYYMMDD".
- int non_call_end_date

  Non-call end date, format "YYYYMMDD".
- int auction_call_date

  Auction call date, format "YYYYMMDD".
- int reinvestment_end_date

  Reinvestment end date, format "YYYYMMDD".
- int maturity_date

  Deal maturity date, format "YYYYMMDD".
- int num_pay_periods

  Indicates how many valid determination/payment dates have been populated.
- int determination_dates [MAX_PERIODS]

  Determination dates, format "YYYYMMDD".

- int pay_dates [MAX_PERIODS]

  Pay dates, format "YYYYMMDD".
- int next_callable_date

  Next callable date, format "YYYYMMDD".
- int EOD_date

  EOD date, format "YYYYMMDD".
- int redemption_date

  redemption date, format "YYYYMMDD".

21.11.1 Detailed Description

New feature Subject to change

See Also

get_cdo_date_info()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.12 CDO_TEST_FLOW Struct Reference

This struct stores the test projection information.

```
#include <wsa.h>
```

Data Fields

- int size

  Size of the projection.
- int date [MAX_PERIODS]

  Date of the test.
- double threshold_pct [MAX_PERIODS]

  Test threshold.
- double actual_pct [MAX_PERIODS]

  Calculated percentage value.
- int result [MAX_PERIODS]

  Test result.
- double cure_amount [MAX_PERIODS]

  Test remaining cure amount.
- double idt_buy [MAX_PERIODS]

  Test Interest Diversion Test BUY.

21.12.1 Detailed Description

New feature Subject to change

See Also

get_cdo_test_flow()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.13 CDO_TEST_INFO Struct Reference

This struct stores CDO test information.

```
#include <wsa.h>
```

Data Fields

- char name [50]

  Test name. For IC and PV test it is an tranche name. For user defined test, it is an defined name.
- short level

  Test level.
- TEST_TYPE test_type

  Type of the test.
- double threshold_pct

  Test threshold.
- double actual_pct

  Calculated percentage value.
- BOOLYAN result

  Test result.
- double tr_pct

  Trustee report percentage value.
- BOOLYAN tr_result

  Trustee report test result.
- char test_var [25]

  The test script variable specified as part of the deal model.

21.13.1 Detailed Description

New feature Subject to change

See Also

get_cdo_test_info()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.14 CHINA_BOND_INFO Struct Reference

Data Fields

- char tranche_name [20]

  The tranche name.
- char tranche_code [11]

  The tranche code.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.15 CMM_CUSTOM_ECONOMIC_DATA Struct Reference

Data Fields

- double **macro_data** [13][40]
- bool **macro_data_set** [13]
- double **ir_data** [19][40]
- bool **ir_data_set** [19]

The documentation for this struct was generated from the following file:

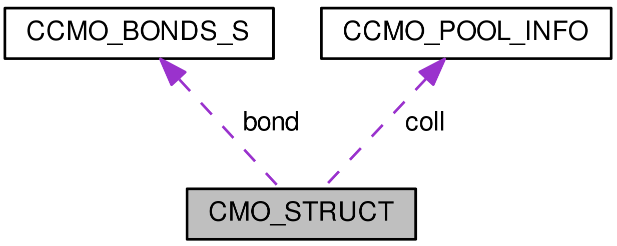
- include/ccmo.h

21.16 CMO_STRUCT Struct Reference

This is the primary structure used to pass information to and from the WSA API.

`#include <ccmo.h>`

Collaboration diagram for CMO_STRUCT:



Data Fields

- char dealid [20]
- char bondid [20]
- BOOLYAN actual_coll
- char next_pay_date [10]
- char settlement_date [10]
- char orig_settlement_date [10]
- char first_pay_date [10]
- char first_projected_date [10]
- short num_bonds
- short num_colls
- short periodicity
- CCMO_POOL_INFO coll
- CCMO_BONDS_S bond
- double deprecated_psa

  Deprecated please use set_prepayment_ex() or callback function.
- double deprecated_cpr

Deprecated please use set_prepayment_ex() or callback function.

- double deprecated_smm [MAX_PERIODS]

  Deprecated please use set_prepayment_ex() or callback function.

- double coll_cash [MAX_PERIODS]
- double coll_bv [MAX_PERIODS]
- double deprecated_rate1 [MAX_PERIODS]

  Obsolete field.

- double deprecated_rate2 [MAX_PERIODS]

  Obsolete field.

- double principal [MAX_PERIODS]
- double interest [MAX_PERIODS]
- double balance [MAX_PERIODS]

21.16.1 Detailed Description

See Also

- open_deal_ex()
- run_deal_ex()
- close_deal_ex()

21.16.2 Field Documentation

21.16.2.1 BOOLYAN CMO_STRUCT::actual_coll

Required. 0, 1 or 2.

Value	Meaning
0	bucketed collateral will be used. The bucketing criteria greatly speeds up the processing time while maintaining a high degree of accuracy.
1	cashflows will be projected using individual pieces of collateral.
2	cashflows will be projected using individual pieces of collateral and additional collateral information from partial deal update for requested update date, if available. (CHS only)

Note

Value should be set before calling open_deal_ex().

21.16.2.2 double CMO_STRUCT::balance[MAX_PERIODS]

Outstanding balance for the bond requested (or first bond in deal).

Note

Value set by WSA API on run_deal_ex().

21.16.2.3 CCMO_BONDS_S CMO_STRUCT::bond

Bond information. If specify bond.id with "EXACT", open deal would check the bondif(above) was set correctly, if bondid is valid, the information will be for that bond, otherwise open deal would fail. If not specify bond.id with "EXACT", and the bondid (above) was set, the information will be for that bond, otherwise it will be for the first bond in the deal.

Note

Value set by WSA API on open_deal_ex().

21.16.2.4 char CMO_STRUCT::bondid[20]

Optional. This is the bond ID. Set this before opening the deal if you want to return information about a specific bond. If no specific bond is needed, set this to null ('\0'), and information about the first bond will be returned (in CMO_STRUCT::bond).

Note

Before calling open_deal_ex(), if CCMO_BONDS_S::id of CMO_STRUCT::bond is set to "EXACT", bondid must be set to a valid value.

21.16.2.5 CCMO_POOL_INFO CMO_STRUCT::coll

Deal-level collateral information. Coupons, loan age, etc. are weighted averages.

Note

Value set by WSA API on open_deal_ex().

21.16.2.6 double CMO_STRUCT::coll_bv[MAX_PERIODS]

Total collateral bond value for the deal. Period 0 should be set to -1 before

Note

Value set by WSA API on run_deal_ex().

each run when amortizing collateral, and to the actual values to used when not using the collateral amortization.

Note

Value set by WSA API on run_deal_ex().

Value should be set by user after open_deal_ex() returns and before calling run_deal_ex().

21.16.2.7 double CMO_STRUCT::coll_cash[MAX_PERIODS]

Total collateral cashflows for the deal. This is only set when not using the collateral amortization.

```
@note Value set by WSA API on run_deal_ex().

 Value should be set by user after open_deal_ex() returns and before calling run_deal_ex().
```

21.16.2.8 char CMO_STRUCT::dealid[20]

Required. This is the deal ID. Set this before opening the deal.

Note

Value should be set before calling open_deal_ex().

21.16.2.9 char CMO_STRUCT::first_pay_date[10]

The first date of cashflows from origination.

Note

Value set by WSA API on open_deal_ex().

21.16.2.10 char CMO_STRUCT::first_projected_date[10]

Optional.If set to ("1"), the deal opened based on as of date and the bond's period-0 payment date.

21.16.2.11 double CMO_STRUCT::interest[MAX_PERIODS]

Interest payments for the bond requested (or first bond in deal).

Note

Value set by WSA API on run_deal_ex().

21.16.2.12 char CMO_STRUCT::next_pay_date[10]

```
mm/dd/yy
```

The first payment date (period 1 cashflows) based on the date the deal is opened as of (see settlement_date member).

Note

Value set by WSA API on open_deal_ex().

21.16.2.13 short CMO_STRUCT::num_bonds

The number of bonds in the deal. This includes paid-off bonds.

Note

Value set by WSA API on open_deal_ex().

21.16.2.14 short CMO_STRUCT::num_colls

The number of pieces of collateral. Paid-off collateral may not be included.

Note

Value set by WSA API on open_deal_ex().

21.16.2.15 char CMO_STRUCT::orig_settlement_date[10]

The original settlement date for the deal.

Note

Value set by WSA API on open_deal_ex().

21.16.2.16 short CMO_STRUCT::periodicity

The number of payments per year.

Note

Value set by WSA API on open_deal_ex().

21.16.2.17 double CMO_STRUCT::principal[MAX_PERIODS]

Principal payments for the bond requested (or first bond in deal).

Note

Value set by WSA API on run_deal_ex().

21.16.2.18 char CMO_STRUCT::settlement_date[10]

```
mm/dd/yy
```

Optional. The status date for the deal to be opened and projections run. The deal will be opened as of that month and year. If set to NULL ('\0')or not set, the deal will be opened with the most recent data.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.17 COLLAT_EXTEND_INFO Struct Reference

This structure used for COLLAT_ASSUMP_CB_EX to store extended loan info.

```
#include <ccmo.h>
```

Data Fields

- double original_coupon

  The coupon at issuance.
- int issue_age

  The age of an loan at issuance.
- int seasoning

  The period between pool group at issuance and next payment.
- int aft_index_of_agencyname

  The AFT related agency name index. Could be 0-17.
- int curr_date_baloons

Indicate if balloons from cutoff date. Could be 0 or 1. 1 indicate balloons from cutoff date.

- int curr_date_amtlock

    Indicate if amortization lockout from cutoff date. Could be 0 or 1. 1 indicate amort lockout from cutoff date.

- int unmod_amtlockout

    Amortization Lockout at issuance.

- int inv_term

    The remaining term of the loan.

- double aft_balance_factor

    The ratio of active assets over all assets in this closing pool.

- double arm_unnmod_lifecap

    The life cap at issuance.

- double arm_unmod_periodic_cap

    The periodic cap at issuance.

- int arm_orig_rate_rperiod

    The rate reset period at issuance.

- int arm_next_rate_reset

    Next rate reset period.

- int issuer_type

    issue type for whole loan.

21.17.1 Detailed Description

See Also

install_collat_assump_cb_ex()

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.18 COUPON_INFO Struct Reference

coupon information.

```
#include <wsa.h>
```

Data Fields

- short coupon_type

    Coupon type, must be one of WHOLE_LOAN_COUPON_TYPE.

- double gross_coupon

    Gross coupon of the loan.

- double net_coupon

    Net coupon of the loan.

- int index

    The market index used to calculate the coupon. Available values: INDEX_TYPE_EX and INDEX_TYPE.

- double margin

    Margin rate adding to reference rate.

- short first_rate_reset

    Months to the first rate reset based on the date the deal is opened as of.

- short first_pay_reset

Months to the first payment reset based on the date the deal is opened as of.(SFW only)

- short reset_freq

  Months between interest rate resets.
- short pay_reset_freq

  Months between payment resets.(SFW only)
- double periodic_cap

  Maximum absolute interest rate change per period. i.e., 2.0% means absolute rate change is at most 2.0%.(SFW only)
- double cap

  Maximum interest rate for this loan.
- double floor

  Minimum interest rate for this loan.
- short day_count
- short pay_freq

  Payment frequency, must be one of PAYMENT_FREQUENCY.
- int conversion_date_1

  First conversion date after origination. format "YYYYMMDD".
- int conversion_date_2

  Second conversion date after origination. format "YYYYMMDD".
- int conversion_date_3

  Third conversion date after origination. format "YYYYMMDD".
- int conversion_date_final

  Final conversion date after origination. format "YYYYMMDD".
- double conversion_coupon

  Converted coupon of the loan.
- double conversion_margin_1

  First converted Margin rate adding to reference rate.
- double conversion_margin_2

  Second converted Margin rate adding to reference rate.
- double conversion_margin_3

  Third converted Margin rate adding to reference rate.
- double conversion_margin_final

  Final converted Margin rate adding to reference rate.

21.18.1 Detailed Description

New feature Subject to change

21.18.2 Field Documentation

21.18.2.1 short COUPON_INFO::day_count

Calender must be one of

- DAYCOUNT_ACTUAL_360
- DAYCOUNT_ACTUAL_365
- DAYCOUNT_ACTUAL_ACTUAL
- DAYCOUNT_30_360
- DAYCOUNT_30_365

- DAYCOUNT_30_360E

The documentation for this struct was generated from the following file:

- include/wsa.h

21.19 DEAL_ACCOUNT_INFO Struct Reference

This structure stores information about deal accounts (reserve, insurance, liquidity) in SFW deals.

`#include <wsa.h>`

Data Fields

- char id [11]

  Account ID.
- short acct_type

  One of the enums from DEAL_ACCOUNT_TYPES.
- char currency
- double orig_balance

  Original balance.
- double orig_max

  Original max balance.
- double current_balance

  Current balance.
- double current_max

  Current max balance.
- short day_count

  Calendar convention code.
- short index

  Code for reference rate.
- double margin

  Margin rate adding to reference rate.
- short fee_type

  Index of fee type.
- BOOLYAN is_default

  If current account is default.
- int default_date

  Default rate if the account is default.
- char insurer [49]

  Name of the insurer.
- char label [31]

  Reserve for future usage.

21.19.1 Detailed Description

See Also

get_deal_account_avail()

21.19.2 Field Documentation

21.19.2.1 char DEAL_ACCOUNT_INFO::currency

Currency code, is one of:

ASCII	Value	Meaning
69	'E'	EUR, Euros
70	'F'	CHF, Swiss Francs
75	'K'	SEK, Swedish Kronors
76	'L'	GBP, British Pounds
85	'U'	USD, US Dollars
82	'R'	RUB, Russian Rubles
78	'N'	NOK, Norwegian krone
68	'D'	DKK, Danish krone
74	'J'	JPY, Japanese Yen
65	'A'	AUD, Australian Dollar
72	'H'	HKD, Hong Kong Dollar
89	'Y'	CNY, Chinese Yuan
67	'C'	CAD, Canadian Dollar
90	'Z'	Other,for non-mapped currency

The documentation for this struct was generated from the following file:

- include/wsa.h

21.20 DISTRESSED_PROPERTY_RECOVERY Struct Reference

distressed property recovery information.

```
#include <wsa.h>
```

Data Fields

- int recovery_lag

  The number of months from the default event on which the recovery payment is made.
- int inflation_start_period

  The snapshot period (typically months from the start of the scenario) on which inflation starts.
- double inflation_rate

  The annual inflation rate commencing from the inflation start period.
- double distressed_property_value

  Specified at the loan level, this is the distressed value of the collateral for the loan.
- double variable_foreclosure_cost

  Specifies a portion of the property value which will comprise part of the foreclosure costs.
- double fixed_foreclosure_cost

  A fixed amount which will comprise part of the foreclosure costs.

21.20.1 Detailed Description

New feature Subject to change

See Also

set_distressed_property_recovery()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.21 DPD_ASSUMPTION Struct Reference

Assumption for Default Probability Distribution.

`#include <wsa.h>`

Data Fields

- short distribution

  The default/loss probability distribution to be applied to the deal. Should be one of enum DPD_DISTRIBUTION_TY-PE.
- short scenario_type

  Scenario type, 0 for defaults and 1 for losses.
- double mean

  Mean for the probability distribution, required when the distribution type is lognormal or inverse normal.
- double standard_deviation

  Standard deviation for the probability distribution, required when the distribution type is lognormal or inverse normal.
- BOOLYAN use_milan_aaa_ce

  If TRUE then use the Milan Aaa CE input, use Standard Deviation if otherwise.
- double milan_aaa_ce

  Editable when the distribution type is lognormal and scenario type is losses.
- BOOLYAN discounted_recoveries

  Editable when the scenario type is losses.
- double revolving_default_factors [MAX_PERIODS]

  Revolving default factor.
- short num_scenarios

  Number of scenarios for the simulation to run. The maximum of num_scenarios must not be greater than 1000.
- BOOLYAN use_revolving_def_timing

  TRUE if users want to use separate default timing curve for revolving assets. Call set_dpd_revolving_default_timing() to set the revolving default timing curve.
- short rating_cap_primary

  Primary proposed rating cap.
- short rating_cap_surveillance

  Surveillance proposed rating cap.

21.21.1 Detailed Description

New feature Subject to change

The documentation for this struct was generated from the following file:

- include/wsa.h

21.22 DPD_RESULT Struct Reference

Result for Default Probability Distribution Simulation.

`#include <wsa.h>`

Data Fields

- char tranche [8]

  Name of the tranche.
- double accrued_int

  Accrued interest for the tranche.
- double average_life_average

  Average "average life" of the tranche across all scenario runs.
- double average_life_sd

  Standard deviation the average life of the tranche across all scenario runs.
- double base_proceeds

  Base proceeds of the tranche across all scenario runs.
- double base_settlement_balance

  Base settlement balance of the tranche across all scenario runs.
- double convexity_average

  Average convexity of the tranche across all scenario runs.
- double convexity_sd

  Standard deviation of the convexity of the tranche across all scenario runs.
- double duration_average

  Average duration of the tranche across all scenario runs.
- double duration_sd

  Standard deviation of the duration of the tranche across all scenario runs.
- double modified_duration_average

  Average modified duration of the tranche across all scenario runs.
- double modified_duration_sd

  Standard deviation of the modified duration of the tranche across all scenario runs.
- double principal_loss_average

  Average principal loss of the tranche across all scenario runs.
- double principal_loss_sd

  Standard deviation of the principal loss of the tranche across all scenario runs.
- double pv_at_coupon_average

  Average pv at coupon of the tranche across all scenario runs.
- double pv_at_coupon_sd

  Standard deviation of pv at coupon of the tranche across all scenario runs.
- double reimbursed_loss_average

  Average reimbursed loss of the tranche across all scenario runs.
- double reimbursed_loss_sd

  Standard deviation of reimbursed loss of the tranche across all scenario runs.
- double total_cashflow_average

  Average total cashflow of the tranche across all scenario runs.
- double total_cashflow_sd

  Standard deviation of total cashflow of the tranche across all scenario runs.
- double total_interest_average

  Average total interest of the tranche across all scenario runs.
- double total_interest_sd

  Standard deviation of total interest of the tranche across all scenario runs.
- double total_principal_average

  Average total principal of the tranche across all scenario runs.
- double total_principal_sd

  Standard deviation of total principal of the tranche across all scenario runs.
- double price_average

Average price of the tranche across all scenario runs.

- double price_sd

  Standard deviation of price of the tranche across all scenario runs.
- double yield_average

  Average yield of the tranche across all scenario runs.
- double yield_sd

  Standard deviation of yield of the tranche across all scenario runs.
- double dm_average

  Average discounted margin of the tranche across all scenario runs.
- double dm_sd

  Standard deviation of the discounted margin of the tranche across all scenario runs.
- char currency [4]
- double balance

  Balance of the tranche.
- char rating_target [5]

  Target rating of the tranche.
- double wal

  Weighted average life of the tranche.
- double expected_loss_el

  Expected loss of the tranche in percentage.
- double expected_loss_mm

  Moody's metric for the given expected loss.
- char expected_loss_rating [6]

  Corresponding rating for the given expected loss.
- double principal_interest_dp

  Probability of default on either principal or interest in percentage.
- double principal_interest_mm

  Moody's metric for the given probability of default.
- char principal_interest_rating [6]

  Corresponding rating for the given probability of default.
- double principal_loss_dp

  Probability of default on principal only in percentage.
- double principal_loss_mm

  Moody's metric for the given probability of default.
- char principal_loss_rating [6]

  Corresponding rating for the given probability of default.
- char proposed_rating [6]

  Final proposed rating for the tranche.
- double breakeven_probability

  Breakeven probability.

21.22.1 Detailed Description

New feature Subject to change

21.22.2 Field Documentation

21.22.2.1 char DPD_RESULT::currency[4]

Currency of the tranche. Valid Output:

- "USD","GBP","CAD","DEM","UDI",
- "VSM","JPY","CHF","EUR","SDR",
- "ARS","AUD","ATS","BES","BEF",
- "BRL","CLP","CNY","DKK","EGP",
- "FIM","GRD","HKD","ISK","INR",
- "IDR","LUF","MXN","NZD","NOK",
- "PKR","PEN","PHP","RUB","SGD",
- "ZAR","KRW","ESP","SEK","TWD",
- "THB","TRL","UAH","FX1","FX2",
- "DFL","PLN"

The documentation for this struct was generated from the following file:

- include/wsa.h

21.23 DPD_SCENARIO Struct Reference

Scenario for Default Probability Distribution Simulation.

```
#include <wsa.h>
```

Data Fields

- short scenario_num

  Index number for the scenario.
- double default_loss

  Default/loss rate for the scenario.
- double scenario_probability

  Probability for the scenario to occur.
- double cumulative_probability

  Cumulative probability of the scenarios up to the underlying one.

21.23.1 Detailed Description

New feature Subject to change

The documentation for this struct was generated from the following file:

- include/wsa.h

21.24 ESG_CURRENCY_RATE_INPUTS Struct Reference

This structure stores specified inputs to setup ESG specified currency simulation interest rates.

```
#include <wsa.h>
```

Data Fields

- char Currency [4]

  Standard currency code for the interest rate,refer to ISO 4217.
- double ∗ OverrideLiborZCBP

  The array of libor override rates of zero coupon bond price.
- int OverrideLiborZCBPSize

  The number of elements of OverrideLiborZCBP.
- double ∗ OverrideTsyZCBP

  The array of treasury override rates of zero coupon bond price.
- int OverrideTsyZCBPSize

  The number of elements of OverrideTsyZCBP.
- bool UpdateInitialCurveFlag

  Flag to enable or disable initializing ESG curve rates with spot rates.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.25 ESG_MODEL_INPUTS Struct Reference

This structure stores inputs to setup ESG model interest rates.

```
#include <wsa.h>
```

Data Fields

- char ESGbhmPath [1024]

  Path of ESG installer bhm files and dlls.
- char ESGLicenseFilePath [1024]

  ESG license file.
- char ESGCalibrationFilePath [1024]

  ESG calibration file.
- char ESGDataPath [1024]

  Path of ESG installer data, optional.
- char ESGRatesOutputPath [1024]
- ESG_RATE_TYPE ESGRateType

  ESG rates output type.
- int YYYYMMDD

  Simulation date, the format would be yyyymmdd.
- int SimulationPaths

  Num of paths for simulation.
- int Periods

  Periods for simulation.

21.25.1 Detailed Description

See Also

- set_up_ESG_model_interest_rates()

21.25.2 Field Documentation

21.25.2.1 char ESG_MODEL_INPUTS::ESGRatesOutputPath[1024]

ESG rates output Path, optional, if set, just generates ESG output .CSV file under the output path; if not set, WSAAPI would load/set ESG rates automatically when set_metrics_input_ex with OAS enabled.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.26 FFIEC_INPUT_PARAMS Struct Reference

This struct stores FFIEC tests input information.

`#include <wsa.h>`

Data Fields

- FFIEC_TEST_MODE test_mode [7]

  FFIEC scenario test results base on what mode, must be one of the FFIEC_TEST_MODE.
- double input_tsy_yield [7]

  It is only required when FFIEC scenario test_mode is INPUT_MODE, FFIEC results base on input_tsy_yield.
- bool price_include_accrued

  FFIEC results projected_price included accrued interest or not.
- double up_wal_limit

  Upward average life limit, default is 4.0 .
- double down_wal_limit

  Downward average life limit, default is 6.0.
- double price_change_limit

  Price change limit, default is 17.0.

21.26.1 Detailed Description

See Also

get_bond_FFIEC_results()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.27 FFIEC_RESULTS Struct Reference

This struct stores results for FFIEC test.

`#include <wsa.h>`

Data Fields

- bool test_result

  FFIEC test result, true is pass, false is failed.
- double average_life

  Weighted average life.
- double wal_change

  The difference between current mode wal and +0bps mode wal.
- double tsy_yield

  Treasury yield in current mode.
- double pricing_yield

  The yield used for pricing in current mode.
- double projected_price

  The market price based on pricing yield in current mode.
- double percent_change

  The percent change of current mode price based on +0bps mode price.

21.27.1 Detailed Description

See Also

get_bond_FFIEC_results()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.28 FIRST_LOSS_INPUT Struct Reference

Data Fields

- FIRST_LOSS_RUN_MODE first_loss_run_mode

  Run mode. Must be one of enum FIRST_LOSS_RUN_MODE. SFW only.
- BOOLYAN is_percentage

  If TRUE, the output FIRST_LOSS_RESULT.deal_collateral_losses is in percentage. If False, it is in dollar. SFW only.
- short prepayment_type
- double prepayment_rate

  Prepayment rate. SFW only.
- short default_type
- double default_rate

  Default rate.
- double forbearance_rate

  Forbearance rate. Only available for Student Loan. SFW only.
- double deferment_rate

  Deferment rate. Only available for Student Loan. SFW only.
- FIRST_LOSS_THRESHOLD first_loss_threshold

  The first loss threshold. Must be one of FIRST_LOSS_THRESHOLD enum.
- double principal_payment_rate

  Principal payment rate. Only available for Credit Cards. SFW only.
- double monthly_purchase_rate

  Monthly purchase rate. Only available for Credit Cards. SFW only.

- double portfolio_yield

  Portfolio yield. Only available for Credit Cards. SFW only.
- double loss_rate

  Loss rate. Only available for Credit Cards. SFW only.
- double prin_loss_serverity

  prin_loss_serverity. Principal loss percentage. SFW only.
- short run_mode_cc
- BOOLYAN is_true

  If TRUE, the calculation will take into account tranches that defer then pik. CDOnet only.

21.28.1 Field Documentation

21.28.1.1 short FIRST_LOSS_INPUT::default_type

Type of default curve. Must be one of:

- DEFAULT_CURVE_CDR - Constant Default Rate(CDR): The Constant Default Rate is the percentage of the mortgages/loans
- DEFAULT_CURVE_SDA - Standard default curve: Measuring for defaults in the residential mortgage market
- DEFAULT_CURVE_SEASONED_CDR - The Constant Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the year
- DEFAULT_CURVE_SEASONED_MDR - The Monthly Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the month
- DEFAULT_CURVE_PCT - The PCT is similar to the MDR curve except that defaults are applied each month to the period 0 balance of the loan, rather than the

21.28.1.2 short FIRST_LOSS_INPUT::prepayment_type

Type of prepayment curve. SFW only. Must be one of:

- PREPAY_CURVE_PSA - The PSA Standard Prepayment Assumptions rate specifies an annual prepayment percentage as a function of the
- PREPAY_CURVE_SMM - The Single Monthly Mortality rate is the percentage of the mortgages/loans outstanding at the beginning of the month
- PREPAY_CURVE_CPR - The Constant Prepayment Rate is the percentage of the mortgages/loans outstanding at the beginning of a year assumed to terminate during the year.
- PREPAY_CURVE_CPY - The Constant Prepayment Yield rate is equivalent to the Constant Prepayment Rate(CPR) except that it assumes prepayment only happens after contractual lockout and yield maintenance period.
- PREPAY_CURVE_HEP - The Home Equity Prepayment rate is a measure of prepayments for closed-end, fixed rate HEL loans.
- PREPAY_CURVE_ABS - Asset-Backed Securities(ABS):

21.28.1.3 short FIRST_LOSS_INPUT::run_mode_cc

SFW only. Must be one of: CREDIT_RATE_MODES { CC_PRIN_PAY_MODE, CC_PURCHASE_MODE, CC_YIELD_MODE, CC_LOSS_MODE, CC_DEFAULT_MODE, CC_RECOVERY_MODE,

The documentation for this struct was generated from the following file:

- include/wsa.h

21.29 FIRST_LOSS_RESULT Struct Reference

Data Fields

- double run_mode_rate

  The rate of the input run mode. SFW only.
- double deal_collateral_losses

  If true it is mega. If false it is not. SFW only.
- int first_loss_date

  First loss date, format "YYYYMMDD". SFW only.
- double default_rate

  Default rate. CDOnet only.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.30 GLOBAL_REINVESTMENT_ASSET_INFO Struct Reference

This struct stores asset information for global reinvestment.

```
#include <wsa.h>
```

Data Fields

- short num_allocation_ratio

  The length of the vector that has been passed by users to allocation_ratio.
- double allocation_ratio [MAX_PERIODS]

  Allocation ratio of the asset in the reinvestment portfolio.
- char currency [4]

  Standard currency code of the asset,refer to ISO 4217.
- short term

  Term of the asset.
- short pay_freq

  Payment frequency of the asset, refer to PAYMENT_FREQUENCY.
- short day_count
- BOOLYAN io

  Interest only.
- BOOLYAN po

  Principal only.
- BOOLYAN adjustable

  Indicates if the asset if floating or not.
- double coupon

  Coupon of the asset.
- double leverage

  Leverage for coupon calculation.
- short index

  Reference index for coupon.
- short num_margin

  This field indicates how many payment date margins have been input by the user. If the value is set to 0, API will use the value in margin[0] as the fixed margin for the asset.

- double margin [MAX_PERIODS]

  Coupon margins used for the corresponding payment dates of an asset. If the update date of the deal is 01/Jan/2013, and the asset pays quarterly, then margin[0] will be the margin used to calculate interest for 01/Apr/2013, margin[1] for 01/Jul/2013.
- double cap

  Coupon cap.
- double floor

  Coupon floor.
- double libor_floor

  Libor floor.
- char first_reset [9]

  First reset date, of format "YYYYMMDD".
- short reset_freq

  Rate reset frequency.
- double market_price

  Market price of the asset, an input of 99 means the market price of the asset is 99.
- double recovery_rate

  Recovery rate of the asset, an input of 0.7 means the recovery rate is 70%.
- BOOLYAN bank_loans

  CBO Asset Check Bank Loans.
- short seniority

  Seniority of the asset, refer to enum ASSET_SENIORITY.
- char country [4]

  The three-character country codes, please refer to ISO 3166-1 alpha-3.
- char moodys_rating [5]
- char company_name [50]

  Company name.

21.30.1 Detailed Description

New feature Subject to change

See Also

set_global_reinvestment()

21.30.2 Field Documentation

21.30.2.1 short GLOBAL_REINVESTMENT_ASSET_INFO::day_count

Day count convention, must be one of:

- DAYCOUNT_ACTUAL_360
- DAYCOUNT_ACTUAL_365
- DAYCOUNT_ACTUAL_ACTUAL
- DAYCOUNT_30_360
- DAYCOUNT_30_365

21.30.2.2 char GLOBAL_REINVESTMENT_ASSET_INFO::moodys_rating[5]

Reported Moodys Rating, should be one of:

- "Aaa", "Aa1", "Aa2", "Aa3", "A1", "A2", "A3",
- "Baa1", "Baa2", "Baa3", "Ba1", "Ba2", "Ba3", "B1", "B2", "B3",
- "Caa1", "Caa2", "Caa3", "Ca", "C",
- "D", "LD", "WR",

The documentation for this struct was generated from the following file:

- include/wsa.h

21.31 GLOBAL_REINVESTMENT_INFO Struct Reference

This struct stores information of global reinvestment setting.

```
#include <wsa.h>
```

Data Fields

- BOOLYAN auto_reinv_end_date_override

  If true, the reinvestment end date will be set to "6 months before maturity" regardless of the user input for auto_reinv-_end_date.
- char auto_reinv_end_date [9]

  Auto reinvestment end date, of format "YYYYMMDD".
- short term_setting

  Reinvestment term settings, refer to enum REINV_TERM_SETTING_TYPE.
- double ppy_pct_in_rp

  Percentage of prepayment that will be used for reinvestment during reinvestment period.
- double ppy_pct_after_rp

  Percentage of prepayment that will be used for reinvestment after reinvestment period.
- double sched_pct_prin_in_rp

  Percentage of scheduled principal that will be used for reinvestment during reinvestment period.
- double sched_pct_prin_after_rp

  Percentage of scheduled principal that will be used for reinvestment after reinvestment period.
- double rec_pct_in_rp

  Percentage of recovery that will be used for reinvestment during reinvestment period.
- double rec_pct_after_rp

  Percentage of recovery that will be used for reinvestment after reinvestment period.
- double curr_pct_prin_cash_in_rp

  Percentage of money in current principal account that will be used for reinvestment during reinvestment period.
- double curr_pct_prin_cash_after_rp

  Percentage of money in current principal account that will be used for reinvestment after reinvestment period.
- BOOLYAN use_global_reinvestment_override

  True if the user wants to use deal periodicity/currency/benchmark/reset period in the asset pool setting.
- short num_global_libor_floor

  The length of the vector that has been passed by users to global_libor_floor, 0 means not use global libor floor for assets.
- double global_libor_floor [MAX_PERIODS]

  Libor floor vector for all assets.

21.31.1 Detailed Description

New feature Subject to change

See Also

set_global_reinvestment()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.32 HECM_INFO Struct Reference

Data Fields

- short payment_plan

  The payment plan (option) of HECM. Need to be one type of HECM_PAYMENT_PLAN.
- double loan_balance

  HECM loan balance.
- double MIP

  Mortgage Insurance Premium: a cost for FHA mortgage insurance charged based on MCA.
- double servicing_amount

  Fee charged by lenders or lenders' agents.
- short draw_term

  Term in months for draw_amount.
- double draw_amount

  Amount of additional money drawn during the draw period, which occurs at the beginning of the month.
- double avail_LOC

  A credit limit that a borrower can draw.
- double max_claim_amount

  The lesser of a home's appraised value or the maximum loan limit that can be insured by FHA.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.33 LICENSE_INFO Struct Reference

This structure contains license realted information.

```
#include <ccmo.h>
```

Data Fields

- char **user_id** [100]
- char **expiration_date** [25]
- char **feature_name** [255]

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.34 LOAN_PRICING_INPUT Struct Reference

price_loan_ex input information.

```
#include <wsa.h>
```

Data Fields

- PRICING_ANCHORS anchorType

  Enum for the pricing anchors.
- double anchor_value

  Value for the related pricing type.
- short index

  This would be used when anchorType is PRICE and DM, can be null or one of INDEX_TYPE or INDEX_TYPE_EX.
- short day_count

  Calendar setting for discounting, need to be "DAYCOUNT", by default it is 30/360.

21.34.1 Detailed Description

New feature Subject to change

See Also

price_loan_ex()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.35 MARKIT_BOND_CASHFLOW Struct Reference

This structure is used to provide user with bond cashflow. It has projected cashflow as well as possibly one historical cashflow some 0 delay bonds are entitled to. It includes number of cashflow points as well as specific bond cashflow dates. Previously only projected cashflows were available. However some 0 delay floater bond owners might also be entitled to one historical payment. It will be reported in this structure.

```
#include <ccmo.h>
```

Data Fields

- int size

  Size of the cashflow including element of 0 representing balance at the settlement date.
- int ∗ hist_flag
- int ∗ dates

  Array of CCYYMMDD dates. Date of actual bond payment. The first period stores the last payment date of the current deal update.
- int ∗ accrual_begin_dates

  The beginning date that is used to calculate accrual interest for given period.
- double ∗ balance

  Balance at the end of the period.
- double ∗ interest

  Actual interest paid in the given period.

- double ∗ principal

  Actual principal paid in the given period, including scheduled principal, prepayment and principal recoveries.
- double ∗ principal_writedown

  Principal loss due to default. For projected cash flow, this value equals to previous balance minus the sum of the given period balance and principal.
- double ∗ cash

  Sum of interest and principal.
- double ∗ rate

  Rate from which theoretical interest was calculated. For an adjustable rate bond, this rate is either the smaller number uncapped rate and cap rate or the larger number of uncapped rate and floor rate. Also, this field is subject to increase/decrease cap if applicable.
- double ∗ flt_index

  Float index rate. This is a reference rate(like Libor/ Treasury/ COFI/ prime) that the bond refers to. This rate is specified in.
- int start_index_as_per_settle_date
- int reserve_size

  Field reserves for internal use. User shall not change it.
- int ∗ theoretical_dates

  This is the theoretical date for cashflow. It ignores the weekends and holidays. In MARKIT_BOND_INFO, bus_rules indicates the actual accrual beginning date if the theoretical date a non-business day.
- int ∗ theoretical_accrual_begin_dates

  Theoretically, the CCYYMMDD dates where accrual begins. This is the theoretical accrual begin date for calculating the interest. It ignores the weekends and holidays. In MARKIT_BOND_INFO, bus_rules indicates the actual accrual beginning date if the theoretical date a non-business day.
- double ∗ interest_due

  Due interest will be collected. Its value equals to the production of balance from the previous period.
- double ∗ interest_shortfall

  This field is the difference between Interest and Interest Due.
- int ∗ accrual_days

  Number of accrual days.
- double ∗ uncapped_rate

  Rate uncapped. It is the sum of margin rate and reference index rate from the last rate reset day.

21.35.1 Detailed Description

Keep in mind that pointers of this structure will be invalidated once a new structure for the same deal is retrieved from the API.

See Also

get_bond_flow_ex1()

21.35.2 Field Documentation

21.35.2.1 int∗ MARKIT_BOND_CASHFLOW::hist_flag

Indicates if the cashflow is historical or projected

Value	Meaning
0	projected payment
1	historical payment

-1	this is historical payment for which history data is not available. Only date of the payment is reported. Contact your Moody's Analytics representative for historical data

21.35.2.2 int MARKIT_BOND_CASHFLOW::start_index_as_per_settle_date

Start index as per settle date

- number of months between trade settlement date (MARKIT_DEAL_INFO::trade_settlement_date) and bond accrual begin date (MARKIT_BOND_CASHFLOW::accrual_begin_dates[1])
- 0 if trade settlement date is the same or earlier than bond accrual begin date

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.36 MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE Struct Reference

This structure is a variation of structure MARKIT_BOND_CASHFLOW.

```
#include <ccmo.h>
```

Data Fields

- int size

  Size of the cashflow including element of 0 representing balance at the settlement date.
- int hist_flag [MAX_PERIODS]
- int dates [MAX_PERIODS]

  Array of CCYYMMDD dates. Date of actual bond payment.
- int accrual_begin_dates [MAX_PERIODS]

  The beginning date that uses to calculate accrual interest.
- double balance [MAX_PERIODS]

  Balance at the end of the period.
- double interest [MAX_PERIODS]

  Actual interest paid in given period.
- double principal [MAX_PERIODS]

  Actual principal paid in the given period, including schedule principal, prepayments and principal recoveries.
- double principal_writedown [MAX_PERIODS]

  Principal loss due to default. For projected cash flow, this value equals to the previous balance minus the sum of the given balance and the given principal.
- double cash [MAX_PERIODS]

  Sum of interest and principal.
- double rate [MAX_PERIODS]

  Rate from which theoretical interest was calculated. For an adjustable rate bond, this rate is either smaller number of uncapped rate and cap rate or larger number of uncappeed rate and floor rate. This field is also subject to increase/decrease cap, if it is applicable.
- double flt_index [MAX_PERIODS]

  Float index rate. This is a reference rate(ie., Libor/ Treasury/ Cofi/ Prime) that the bond refers to. This rate is specified in scenario from the deal.
- int start_index_as_per_settle_date
- int theoretical_dates [MAX_PERIODS]

Theoretically, the CCYYMMDD dates. This is the theoretical date for cashflow, ignoring the weekends and holidays. In structure MARKIT_BOND_INFO, field bus_rules indicates the actual payment date if the theoretical date is a non-business day.

- int theoretical_accrual_begin_dates [MAX_PERIODS]

  This is theoretical accrual begin date for calculating interest. This is theoretical accrual begin date for calculating interest. It ignores the weekends and holidays. In structure MARKIT_BOND_INFO, field bus_rules indicates the actual accrual begin date if theoretical date is non-business day.
- double interest_due [MAX_PERIODS]

  Interest due. Due interest to be collect, this field value equals to the production of balance from previous period, rate and accrual period(in measurement of year).
- double interest_shortfall [MAX_PERIODS]

  Interest shortfall. This is the difference between Interest and Interest Due.
- int accrual_days [MAX_PERIODS]

  Number of accrual days.
- double uncapped_rate [MAX_PERIODS]

  This field is the sum of margin rate and reference index rate from last rate reset day.

21.36.1 Field Documentation

21.36.1.1 int MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE::hist_flag[MAX_PERIODS]

Flags that are used to indicate if the cash flow is historical or projected. See MARKIT_BOND_CASHFLOW::hist_-flag

21.36.1.2 int MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE::start_index_as_per_settle_date

Start index as per settle date. See MARKIT_BOND_CASHFLOW::start_index_as_per_settle_date

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.37 MARKIT_BOND_INFO Struct Reference

This structure holds individual bond information.

```
#include <ccmo.h>
```

Data Fields

- double orig_balance

  The original balance of the bond when the deal is issued.
- double current_balance

  The outstanding principal of the bond as of the date the deal is opened.
- int next_payment_date

  Next payment date for bond interest and principal.
- double next_payment_beg_balance

  The bond beginning balance on the next payment date.
- double coupon

  The interest rate of the bond as of the date the deal is opened (see CMO_STRUCT::settlement_date).
- int acrual_begin_date

  This is the beginning of the bond accrual period for the current distribution date.

- short delay_days

  The number of days delay between end of interest accrual period and actual bond payment.
- char tranche_name [20]

  Name of the tranche.
- char cusip [10]

  CUSIP id.
- int stated_maturity

  The bond's stated maturity.
- char prin_type [20]

  A description of the principal type of the bond.
- char int_type [20]

  A description of the interest type of the bond.
- short floater_index

  Floaters Only: The market index used to calculate the coupon.Available values: INDEX_TYPE_EX and INDEX_TYP-E.
- double floater_spread

  Floaters Only: This is added to the current index rate∗ multiplier.
- double floater_multiplier

  Floaters Only: This is multiplied by the current index rate to calculate the bond coupon at reset.
- double floater_cap

  Floaters Only: The maximum interest rate for this bond.
- double floater_floor

  Floaters Only: The minimum interest rate for this bond.
- double per_adj_cap

  Floaters Only: The maximum change in coupon for one reset.
- short first_float_per

  Floaters Only: The first period the coupon starts floating.
- short last_float_per

  Floaters Only: The last period the coupon can be adjusted.
- double resume_coupon

  Floaters Only: Coupon rate after the last_float_per.
- short component

  This indicates if the bond is a component bond(positive) or an owner bond(negative). The absolute value of the owner bond and the components component field are the same and this equals the 1-based index of the owner bond in the capital structure. For all other bonds this field is 0.
- short day_count
- char bus_rules
- char markit_currency_code

  Currency of the bond. See DEAL_ACCOUNT_INFO.currency for the available currency types.
- int macr

  If positive than MACR number.
- int notional

  This field is true if the bond has a notional balance i.e. it is an interest only bond.
- int priority_rank

  Priority rank of the tranche within the capital structure of the deal.
- int insurance_flag

  1 if bond is insured, 0 - otherwise.
- int z_bond

  This field is true if the bond is an accrual bond.
- int reset_freq

  The number of months between interest rate resets.

- int coupon_lockout

  Floaters Only: The first period the coupon starts floating.

- int periodicity

  Payments per year.

21.37.1 Detailed Description

This structure is an extension to CCMO_BONDS_S.

See Also

- CMO_STRUCT
- CCMO_BONDS_S
- get_bond_info_by_tranche()
- get_bond_info_by_index()

21.37.2 Field Documentation

21.37.2.1 char MARKIT_BOND_INFO::bus_rules

The rules for determining the payment date (such as next business day if the payment date is not a business day). The interpretation for the field is:

Code	Description	Comments
A	Next Bus Day	If settlement is not a business day, settle on the next business day.
B	Next Bus Day In Month	If settlement is not a business day, settle on the next business day if it is in the current month. If not, settle on the prior business day.
C	Next BD Aft Serv Remit	Next business day after the master servicer remittance date.
P	Prev Bus Day	If settlement is not a business day, settle on the previous business day.
N	No Adjustment	Do not adjust for business days.

21.37.2.2 short MARKIT_BOND_INFO::day_count

The day count rule is used to compute interest. The interpretation for the field is:

Code	Description	Notes
1	Act/360	Actual days in period, 360 day year
2	Act/365	Actual days in period, 365 day year
3	Act/Act	Actual days in period, actual days in year
4	30/360	30 days in month, 360 days in year
5	30E/360	30 days in month, 360 days in year, last day in Feb = 30th.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.38 MARKIT_COLLAT_CASHFLOW Struct Reference

This structure is used to provide user with collateral cashflow. It includes number of cashflow points as well as collateral cashflow dates.

```
#include <ccmo.h>
```

Data Fields

- int size

    Size of the cash flow including element of 0 representing balance at the settlement date.
- int dates [MAX_PERIODS]

    Array of CCYYMMDD dates. The first period stores the last payment date of the current deal update.
- double balance [MAX_PERIODS]

    Balance at the end of the period.
- double sched_principal [MAX_PERIODS]

    Scheduled principal paid.
- double prepayments [MAX_PERIODS]

    Prepayments per period.
- double defaults [MAX_PERIODS]

    Defaults per period with lag.
- double losses [MAX_PERIODS]

    Losses per period.
- double prin_recoveries [MAX_PERIODS]

    This is principal amount recovered when collateral defaults.
- double interest [MAX_PERIODS]

    Interest paid per period.
- double reinvestment [MAX_PERIODS]

    Reinvestment per period.
- double cash [MAX_PERIODS]

    Cash flow per period This field is the sum of interest, scheduled principal, prepayment and principal recoveries from the default collateral.
- double bond_value [MAX_PERIODS]

    Bond value per period, identical to the "balance" field in this structure.
- double prepay_penalties [MAX_PERIODS]

    Prepayment penalty per period. It only applies to that collateral that can be called. For RMSS collateral: This value will be derived from the greater one of the yield maintenance penalty and the penalty based on months of interest or percentage of prepayments.
- int start_index_as_per_settle_date
- double negative_amortization [MAX_PERIODS]

    Negative amortization. This is the absolute value of schedule principal if it is negative. Otherwise it is zero.
- double gross_interest [MAX_PERIODS]

    Field reserves for future use.
- double sched_p_and_i [MAX_PERIODS]

    Scheduled principal and interest. This value is composites of gross interest, scheduled principal and negative amortization.
- double draw_amount [MAX_PERIODS]

    Amount of additional money drawn during the draw period, applicable for HELOCs only, (Home Equity Line of Credit). This amount will be added into performing balance in the next period.
- double total_excess_losses [MAX_PERIODS]

    Field reserves for future use.
- double studentLoanDelayedInterest [MAX_PERIODS]

Field reserves for future use.

- double autoLeaseNetResidualSchedFlow [MAX_PERIODS]

  Field reserves for future use.
- double autoLeaseResidualLoss [MAX_PERIODS]

  Field reserves for future use.
- double po_balance [MAX_PERIODS]

  Balance for PO strips.
- double po_sched_principal [MAX_PERIODS]

  Scheduled principal for PO strips.
- double po_prepayments [MAX_PERIODS]

  Prepayments for PO strips.
- double po_prin_recoveries [MAX_PERIODS]

  Principal recoveries for PO strips.
- double po_losses [MAX_PERIODS]

  Loss for PO strips.
- double premium_loan_balance [MAX_PERIODS]

  Field reserves for future use.
- double excess_interest [MAX_PERIODS]

  Field reserves for future use.

21.38.1 Detailed Description

See Also

get_collateral_flow_ex1()

21.38.2 Field Documentation

21.38.2.1 int MARKIT_COLLAT_CASHFLOW::start_index_as_per_settle_date

Start index as per settle date

- number of months between trade settlement date (MARKIT_DEAL_INFO::trade_settlement_date) and collateral accrual begin date (MARKIT_BOND_CASHFLOW::accrual_begin_dates[1])
- 0 if trade settlement date is the same or earlier than collateral accrual begin date

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.39 MARKIT_DEAL_INFO Struct Reference

This structure contains deal level information of a deal.

```
#include <ccmo.h>
```

Data Fields

- AGENCY_TYPE agency

  Agency type. Must be enum of AGENCY_TYPE.
- char issuer [DEAL_SIZE_OF_ISSUER]

  Issuer ID.
- char deal_name [DEAL_SIZE_OF_DEALNAME]

  Deal name.
- short periodicity

  12 = monthly, 4 = quarterly
- int deal_settlement_date

  Settlement date of the deal.
- int first_pay_date

  First payment after origination.
- int next_pay_date

  Next first payment date from last update.
- char underwriter [DEAL_SIZE_OF_UNDERWRITER]

  Deal underwriter.
- int first_call_date

  The date of first possible call.
- double call_percent

  The percentage of call.
- int update_date

  Deal update date.
- char asset_type [DEAL_SIZE_OF_ASSET_TYPE]
- COLLAT_LEVEL collat_loaded_level

  Collateral level. Must be enum of COLLAT_LEVEL.
- short num_bonds

  Number of bonds in the deal.
- short num_colls

  Number of collaterals in the deal.
- int trade_settlement_date

  Trade settlement date.
- int months_from_last_update_to_settlement

  Number of months from the last update to settlement date.
- int age

  Age.
- char rmbs_type [DEAL_SIZE_OF_ASSET_TYPE]

  Type of rmbs deal, usually be null.

21.39.1 Detailed Description

See Also

get_deal_info()

21.39.2 Field Documentation

21.39.2.1 char MARKIT_DEAL_INFO::asset_type[DEAL_SIZE_OF_ASSET_TYPE]

Underlying asset type. Can be one of the following strings (not limited to):

- Agency
- Agency_CMBS
- Agency_CMBS_FHLMC
- Agency_CMBS_FNMA
- Agency_CMBS_GNMA
- Agency_FHLMC
- Agency_FNMA
- Agency_GNMA
- Agency_HECM
- Auto
- Auto_Lease
- CDO
- CMBS
- CommPaper
- Consumer_Loans
- Credit_Card
- Equip
- FloorPlan
- HECM
- HELOC
- Home_Equity
- Manufactured_Housing
- MBS
- MBS_AgencyRiskShare
- MBS_BuyToLet
- MBS_Prime
- MBS_Subprime
- PPLN
- Private_Label_Agency
- Rec
- SBA
- StructNote

- Student_Loan
- Student_Loan_FFELP
- Student_Loan_Private
- Tax_Liens
- UK_RMBS
- Whole_Loan

The documentation for this struct was generated from the following file:

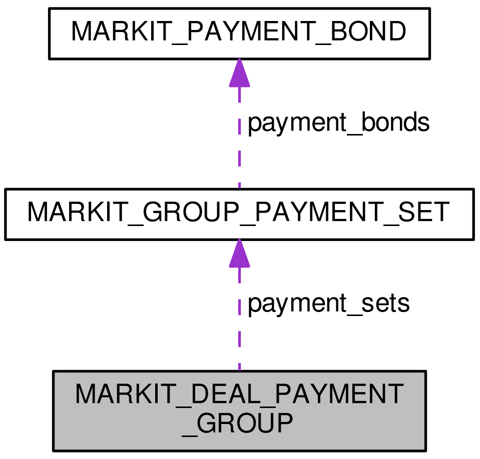
- include/ccmo.h

21.40 MARKIT_DEAL_PAYMENT_GROUP Struct Reference

This structure is the deal level payment group information.

`#include <ccmo.h>`

Collaboration diagram for MARKIT_DEAL_PAYMENT_GROUP:



Data Fields

- char group_name [10]

  The group name.
- double percent

  The payment portion.
- int number_of_sets

  Number of payment sets in the group.
- MARKIT_GROUP_PAYMENT_SET payment_sets [20]

  The payment sets information in the group.

21.40.1 Detailed Description

See Also

get_deal_payment_group()

The documentation for this struct was generated from the following file:

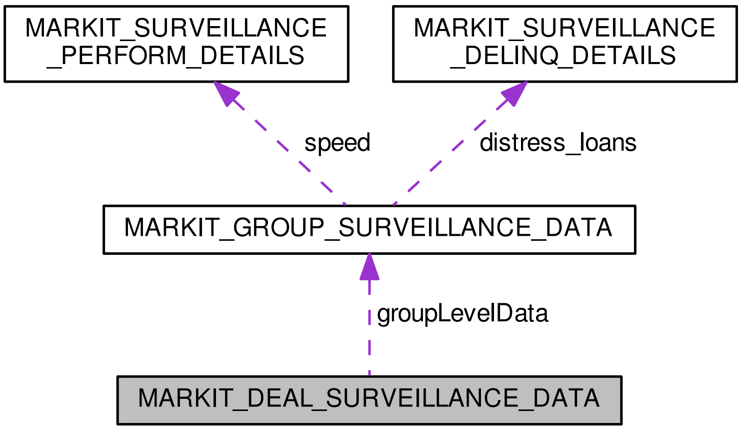
- include/ccmo.h

21.41 MARKIT_DEAL_SURVEILLANCE_DATA Struct Reference

This structure is the deal level surveillance information for a deal.

```
#include <ccmo.h>
```

Collaboration diagram for MARKIT_DEAL_SURVEILLANCE_DATA:


MARKIT_SURVEILLANCE _PERFORM_DETAILS
MARKIT_SURVEILLANCE _DELINQ_DETAILS
speed
distress_loans
MARKIT_GROUP_SURVEILLANCE_DATA
groupLevelData
MARKIT_DEAL_SURVEILLANCE_DATA

Data Fields

- int userReqYYYYMM
  The date of surveillance data that user required for.
- char ticker [DEAL_SIZE_OF_DEALNAME]
  Deal name.
- int reportDate
  The date that this surveillance data is generated for.
- double reqOCAmt
  Required over collateral amount.
- double actOCAmt
  Actual over collateral amount.
- char stepDownTrigger [2]
  Step down trigger. CHS engine only.
- char delinqTrigger [2]
  Delinquency trigger. CHS engine only.

- char cumulLossTrigger [2]

  Cumulative loss trigger. CHS engine only.
- char triggerEvent [2]

  Trigger event. CHS engine only.
- double wala

  Weighted average life age.
- MARKIT_GROUP_SURVEILLANCE_DATA groupLevelData [MAX_COLL_GROUPS]

21.41.1 Detailed Description

See Also

get_deal_surv_data()

Deprecated This structure is deprecated.

21.41.2 Field Documentation

21.41.2.1 MARKIT_GROUP_SURVEILLANCE_DATA MARKIT_DEAL_SURVEILLANCE_DATA::groupLevelData[MAX_-COLL_GROUPS]

Deprecated This field is deprecated..

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.42 MARKIT_GROUP_INFO Struct Reference

This structure includes multiple coupon and balance information on a collateral group.

```
#include <ccmo.h>
```

Data Fields

- double po_coupon

  Coupon for Principal only collaterals.
- double po_balance

  Balance for Principal only collaterals.
- double ioette_coupon

  Coupon for IOette collaterals.
- double excess_coupon

  Excess coupon.
- double ses_coupon

  CHS engine only.

21.42.1 Detailed Description

See Also

get_group_info()

The documentation for this struct was generated from the following file:

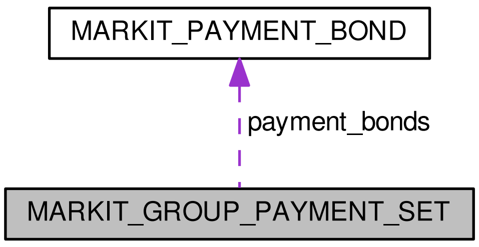
- include/ccmo.h

21.43 MARKIT_GROUP_PAYMENT_SET Struct Reference

This structure contains one set of payment group information.

`#include <ccmo.h>`

Collaboration diagram for MARKIT_GROUP_PAYMENT_SET:



Data Fields

- int number_of_bonds

  Number of bonds in the payment set.
- MARKIT_PAYMENT_BOND payment_bonds [20]

  The payment bonds in information the payment set.

The documentation for this struct was generated from the following file:

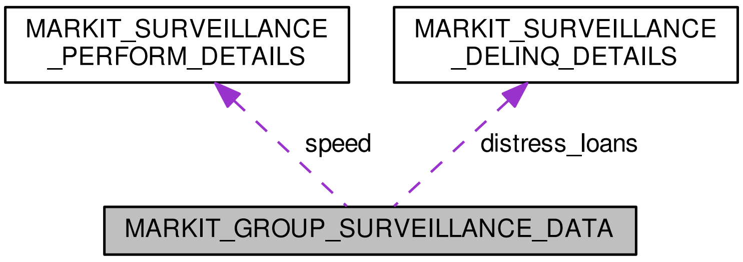
- include/ccmo.h

21.44 MARKIT_GROUP_SURVEILLANCE_DATA Struct Reference

This structure is the collateral group level surveillance information for a deal. This structure is contained in deal level structure MARKIT_DEAL_SURVEILLANCE_DATA.

`#include <ccmo.h>`

Collaboration diagram for MARKIT_GROUP_SURVEILLANCE_DATA:


MARKIT_SURVEILLANCE_PERFORM_DETAILS
MARKIT_SURVEILLANCE_DELINQ_DETAILS
speed
distress_loans
MARKIT_GROUP_SURVEILLANCE_DATA

Data Fields

- int groupId
  Collateral group ID.
- int reportDate
  The date that this surveillance data is generated for.
- double endSchedBalance
  Collateral group balance at end of the schedule.
- double waFico
  Weighted average Fico score.
- double grossWac
  Gross weighted average coupon.
- double netWac
  Net weighted average coupon.
- double waMaturity
  Weighted average maturity.
- double loanCount
  Number of loans in the collateral group.
- double waLtv
  Weighted average LTV.
- double currentPeriodLoss
  The loss on the current period.
- double cumulativeLosses
  Cumulated losses.
- double originalPoolBalance
  The original pool balance.
- double actOCAmt
  Actual over collateral amount.
- double reqOCAmt
  Required over collateral amount.
- double wala
  Weighted average life age.
- MARKIT_SURVEILLANCE_DELINQ_DETAILS distress_loans [SURV_DISTRESS_STATES_NUMBER]
  Delinquent information for distressed loans.
- MARKIT_SURVEILLANCE_PERFORM_DETAILS speed [SURV_PERFORM_DATA_SIZE]
  Prepayment speed and default information.

21.44.1 Detailed Description

Deprecated This structure is deprecated.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.45 MARKIT_HELOC_LOAN_INFO Struct Reference

This structure contains additional information required by HELOC loans.

```
#include <ccmo.h>
```

Data Fields

- AMORTIZATION_TYPE amortization_type
- double draw_limit

  Ratio of the maximum allowable balance over original balance.
- int last_draw_period

  Number of months after origination after which draw is not allowed. 0 if no limit.

21.45.1 Field Documentation

21.45.1.1 AMORTIZATION_TYPE MARKIT_HELOC_LOAN_INFO::amortization_type

Type of amortizations of the loan. Possible values are:

- AMORTIZATION_TYPE_LEVEL_PAYMENT
- AMORTIZATION_TYPE_STRAIGHT_AMORTIZER

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.46 MARKIT_PAYMENT_BOND Struct Reference

This structure contains the bond be payment of group.

```
#include <ccmo.h>
```

Data Fields

- char bondid [10]

  The bond id.
- double percent

  The payment portion.
- BOOLYAN current_rata

  Is C_RATA or not.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.47 MARKIT_PAYMENT_SCHEDULE Struct Reference

This structure contains two arrays in parallel to describe the payment schedule information.

```
#include <ccmo.h>
```

Data Fields

- DAYT pay_date [MAX_PERIODS]

  Dates for the payment to occur.
- double pay_amount [MAX_PERIODS]

  Amount occurs at each pay date above.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.48 MARKIT_POOL_HISTORY_DATA Struct Reference

This structure contains the pool history data.

```
#include <ccmo.h>
```

Data Fields

- int yyyymmdd

  The update date for the history data.
- double val

  The history data.

The documentation for this struct was generated from the following file:

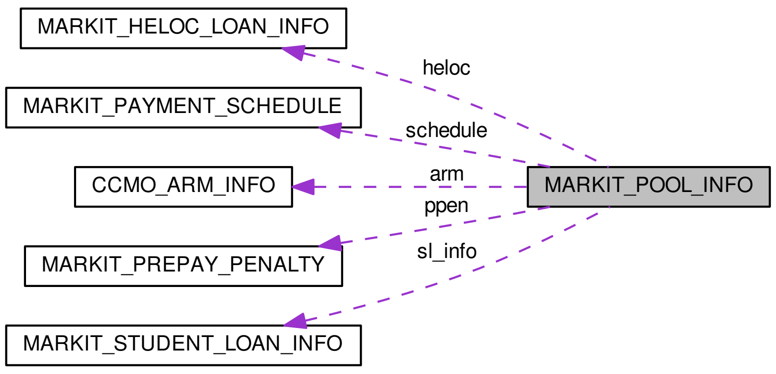
- include/ccmo.h

21.49 MARKIT_POOL_INFO Struct Reference

This structure holds information about collateral.

```
#include <ccmo.h>
```

Collaboration diagram for MARKIT_POOL_INFO:



Data Fields

- int loan_number

  An ordinal for loans inside a deal.
- char id [POOL_ID_LENGTH]

  Identifies the piece of collateral. If reremic, it is the deal-tranche id.
- long number

  For internal use.
- short type

  The type of collateral. Must be enum of POOL_TYPE. This corresponds to MARKIT_POOL_INFO::type_str.
- char type_str [10]

  The description of the type.
- char remic_deal_name [80]

  Remic deal name if reremic, null for a regular pool.
- double pass_through

  The current pass_through rate as of the date the deal is opened. This will be net of trustee and servicing fees. If this is an accumulation of collateral, it will be a weighted average.
- double original_balance

  The original balance of the collateral. If this is an accumulation of collateral, it will only include collateral that is still outstanding.
- double factor

  The factor of the collateral. The current balance is the factor ∗ original_balance.
- double gross_coupon

  The highest gross coupon of the collateral in an agency pool. Irrelevant for non-agencies.
- double wac

  The actual gross coupon of the collateral piece. If this is an accumulation of collateral, it will be a weighted average.
- short term

  The longest original maturity (in months) of the collateral in an agency pool. Irrelevant for non-agencies.
- short wam

  The remaining months until maturity (as of the date the deal is opened). If this is an accumulation of collateral, it will be a weighted average.
- short original_term

  The original term in months. If this is an accumulation of collateral, it will be a weighted average.

- double psa_coupon

  Strip PO: The coupon used by prepayment models (the actual coupon is 0.) Irrelevant otherwise, set to the pass-through rate.
- short wala

  The age of the collateral (as of the date the deal is opened). If this is an accumulation of collateral, it will be a weighted average.
- short balloon_period

  The balloon period.
- short coll_group

  The collateral group this piece belongs to. All collateral in a bucket will belong to the same collateral group.
- double level_pay

  The current level payment for the collateral. If 0, the payment will be calculated.
- double price

  Reserved.
- short prepay_lockout

  The remaining prepayment lockout in months (as of the date the deal is opened).
- short yield_maintain

  Yield maintenance term.
- short prin_lockout

  The remaining principal lockout in months (as of the date the deal is opened).
- short forward_purchase

  The remaining forward purchase months (as of the date the deal is opened).
- short io

  If true(non-zero), collateral is interest-only.
- double current_balance

  Current unpaid balance (as of the date the deal is opened).
- double avg_loan_bal

  The average loan balance.
- double ltv

  Original Loan To Value.
- short fico

  FICO score.
- char servicer_seller [POOL_SERVICER_NAME_LENGTH]

  The servicer/seller for the collateral.
- char delinquency [5]
- char state [POOL_STATE_LENGTH]

  The two-character state code.
- char country [POOL_COUNTRY_LENGTH]

  The three-character country codes, please refer to ISO 3166-1 alpha-3.
- char purpose [POOL_PURPOSE_LENGHT]
- char property_type [POOL_PROPERTY_TYPE_LENGTH]
- char occupancy [POOL_OCCUPANCY_LENGTH]
- char zip [POOL_ZIP_LENGTH]

  Zip code of property.
- short pmi

  Private Mortgage insurance.
- short doc
- short lien_type
- short periodicity

  The number of payments per year.
- short original_balloon_period

Original balloon period.

- short original_prin_lockout

  Original principal lockout term.
- short original_prepay_lockout

  Original prepayment lockout term.
- double ltv_combined

  Loan To Value for all liens.
- double delinq_states [POOL_DELINQ_STATES_SIZE]

  Count of loans belonging to one of 6 delinquency states: current, 30+ days, 60+ days, 90+ days, foreclosed, REO, or terminated. Detailed meanings can be reviewed at POOL_DELINQ_STATES.
- double bankruptcy

  Count of bankruptcy loans.
- double original_pmi_prct

  Original Private Mortgage Insurance percentage.
- short original_prepay_penalty_period

  Original prepayment penalty period.
- double ltv_cur

  Current Loan To Value.
- void ∗ usr_data

  The pointer to store user specific pool level data. Please refer to install_collat_assump_cb() and set_pool_level_user-_data_for_cb() for the usage of this field.
- MARKIT_PAYMENT_SCHEDULE ∗ schedule

  Payment schedule, of type MARKIT_PAYMENT_SCHEDULE. This should either be allocated or set to NULL.
- MARKIT_PREPAY_PENALTY ∗ ppen [MAX_PERIODS]
- CCMO_ARM_INFO ∗ arm

  The structure containing adjustable rate information. This should either be allocated or set to NULL.
- MARKIT_HELOC_LOAN_INFO ∗ heloc

  The pointer to HELOC_INFO structure or 0 if not a HELOC.
- short day_count

  Day count.
- short delay_days

  Delay days.
- MARKIT_STUDENT_LOAN_INFO ∗ sl_info

  Info about student loan.

21.49.1 Detailed Description

This structure is an extension to CCMO_POOL_INFO and CCMO_POOL_INFO_EX.

See Also

- CCMO_POOL_INFO
- CCMO_POOL_INFO_EX
- PAY_POOL_INFO
- get_next_collat()
- get_average_collat()
- get_average_collat_by_bond()
- replace_collateral()

21.49.2 Field Documentation

21.49.2.1 char MARKIT_POOL_INFO::delinquency[5]

The delinquency status of the collateral. The value should be|

Value	Meaning
"0"	POOL_DELINQ_CURRENT
"1"	Delinquent over 30 days
"2"	Delinquent over 60 days
"3"	Delinquent over 90 days
"4"	Foreclosed
"5"	Real estate owned
"6"	Terminated
"7"	Delinquent over 120 days
"8"	Delinquent over 150 days
"9"	Delinquent over 180 days
"10"	Defeasance status
"11"	Non performing matured balloon
"12"	Delinquent over 0 days
"13"	Bankrupt
"14"	Paid off
"15"	Repurchased
"16"	Liquidated
"17"	Closed

21.49.2.2 short MARKIT_POOL_INFO::doc

One of POOL_DOCUM_TYPES:

- POOL_DOCUM_OTHER
- POOL_DOCUM_FULL
- POOL_DOCUM_LIMITED
- POOL_DOCUM_SISA
- POOL_DOCUM_SIVA
- POOL_DOCUM_NINA
- POOL_DOCUM_NO_RATIO
- POOL_DOCUM_NO_DOC
- POOL_DOCUM_ALTERNATIVE
- POOL_DOCUM_UNKNOWN

21.49.2.3 short MARKIT_POOL_INFO::lien_type

One of POOL_LIEN_TYPES:

- POOL_LIEN_TYPE_OTHER
- POOL_LIEN_TYPE_FIRST
- POOL_LIEN_TYPE_SECOND
- POOL_LIEN_TYPE_THIRD

21.49.2.4 char MARKIT_POOL_INFO::occupancy[POOL_OCCUPANCY_LENGTH]

The type of occupancy.

Value	Meaning
1	Primary
2	Secondary
3	Investment
4	Unknown

21.49.2.5 MARKIT_PREPAY_PENALTY* MARKIT_POOL_INFO::ppen[MAX_PERIODS]

Prepayment penalty along pay periods, of type MARKIT_PREPAY_PENALTY. This should either be allocated or set to NULL.

21.49.2.6 char MARKIT_POOL_INFO::property_type[POOL_PROPERTY_TYPE_LENGTH]

The type of property.

Value	Meaning
0	Other
1	Single Family
2	Multi Family
3	Condo
4	PUD
5	Commercial
6	Coop
7	Mobile Home
8	Manufactured Housing
9	Not Available
10	Duplex
11	Triplex
12	Fourplex
13	5+ Units

21.49.2.7 char MARKIT_POOL_INFO::purpose[POOL_PURPOSE_LENGHT]

The purpose of the loan.

Value	Meaning
0	Other
1	Purchase
2	Cash Out Refinance
3	Home Improvement
4	New Construction
5	Rate Term Refinance
6	Not Available

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.50 MARKIT_PREPAY_PENALTY Struct Reference

This structure contains prepayment penalty information.

```
#include <ccmo.h>
```

Data Fields

- short type

  Type of prepayment penalty. MONTHS_INTEREST or PCT_OF_PPY, or 0 for none.
- double multiplier

  multiplier; i.e., 6 for 6 months interest or 6% penalty
- short ym_type

  0 for no yield maintenance, 1 otherwise (for now)
- double scaling

  Used to adjust the prepayment penalty.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.51 MARKIT_STUDENT_LOAN_INFO Struct Reference

This structure holds information about student loan.

```
#include <ccmo.h>
```

Data Fields

- double deferGrossMargin

  Margin in deferment period.
- double deferredInterest

  Total outstanding unpaid accrued interest.
- bool subsidizedInterestDuringDeferment

  Indicates whether or not deferred interest will be subsidized.
- char loanProgramType [40]

  Should be: continuing Education/Graduate/K-12/Medical/Other/Undergraduate.
- STUDENT_LOAN_STATE loanState

  Indicates the status of the loan. Must be enum of STUDENT_LOAN_STATE.
- int remainMonthsInState
- STUDENT_LOAN_REPAY_TYPE repayType

  Indicates the loan is in full deferment of P&I or only deferment of principal. Must be enum of STUDENT_LOAN_REPAY_TYPE.

21.51.1 Field Documentation

21.51.1.1 int MARKIT_STUDENT_LOAN_INFO::remainMonthsInState

Remaining Months to Maturity after entering repayment if the loan in payment status. Remaining Months if the loan in non-payment status for (an aggregate term of any non-payment status: in-school, grace, forbearance, or deferment).

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.52 MARKIT_SURVEILLANCE_DELINQ_DETAILS Struct Reference

This structure contains prepayment penalty information.

```
#include <ccmo.h>
```

Data Fields

- char distress_state [SUVR_DISTRESS_STATE_DESC_LENGTH]

  One of the following states: 'Bankrupt', 'Delinquent', 'Foreclosing', or 'REO'.
- char delinq_state [SUVR_DISTRESS_STATE_DESC_LENGTH]

  If the distress_date is "Delinquent", this field further describe how long the delinquency is. It is either "0 to 29 days", "30 to 59 days", "60 to 89 days", or "90+ days".
- double number_of_loans

  The number of loans.
- double total_current_balance

  The total current balance.
- double loan_percentage

  The percentage of the number of the distressed loans over the total number of loans.
- double balance_percentage

  The percentage of the distressed balance over the total balance.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.53 MARKIT_SURVEILLANCE_PERFORM_DETAILS Struct Reference

This structure contains loan performance, specifically the prepayment speed and default information.

```
#include <ccmo.h>
```

Data Fields

- SURV_PERFORM_MEASURE_TYPE measure

  'CDR' or 'CPR'
- SURV_PERFORM_PERIOD_TYPE period

  '1M' or '3M'
- double value

  Performance data.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.54 MarkitAdcoDefaultModelDials Struct Reference

The fine tune parameters for the ADCO default model. This supplements structure MarkitAdcoTuningParam, and is optional.

```
#include <WSAAdcoProviderApi.h>
```

Data Fields

- double TuneCD

  Tune C-D transition.
- double TuneDC

  Tune D-C transition.
- double TuneDS

  Tune D-S transition.
- double TuneDT

  Tune D-T transition.
- double TuneFICO_Slide

  Tune FICO/Credit Score by sliding left (negative) or right (positive)
- double TuneFICO_Stretch

  Tune FICO/Credit Score by stretching difference between its value and a midpoint. > 1.0 stretches, < 1.0 contracts.
- double TuneHPI_Slide

  Tune Home Price Growth by sliding left (negative) or right (positive)
- double TuneHPI_Stretch

  Tune Home Price Growth by stretching difference between its value and a midpoint. > 1.0 stretches, < 1.0 contracts.
- double TuneMDR

  Scale Default Rate.
- double TuneProbLossTC

  Tune Probability of Loss from C Termination.
- double TuneProbLossTD

  Tune Probability of Loss from D Termination.
- double TuneProbLossTS

  Tune Probability of Loss from S Termination.
- double TuneSATOHat

  Tune Sato hat in bp.
- double TuneSATO_Residual

  Tune SATO residual. Direct multiplier on the SATO residual value.
- double TuneSC

  Tune S-C transition.
- double TuneST

  Tune S-T transition.
- double TuneSeverity

  Scale Severity.
- double TuneSeverityTC

  Tune Loss Severity from C Termination.
- double TuneSeverityTD

  Tune Loss Severity from D Termination.
- double TuneSeverityTS

  Tune Loss Severity from S Termination.
- double TuneWAOLTV_Slide

  Tune Weighted Average LTV by sliding left (negative) or right (positive)
- double TuneWAOLTV_Stretch

  Tune Weighted Average LTV by stretching difference between its value and a midpoint. > 1.0 stretches, < 1.0 contracts.

The documentation for this struct was generated from the following file:

- include/WSAAdcoProviderApi.h

21.55 MarkitAdcoPrepayModelDials Struct Reference

The fine tune parameters for the ADCO prepay model. This supplements structure MarkitAdcoTuningParam, and is optional.

```
#include <WSAAdcoProviderApi.h>
```

Data Fields

- double TuneAge

  Speed-up/Slow-down Aging(range [0,2], default 1.0. Bigger means faster prepays)
- double TuneBurnout

  Tune Burnout(range [0,2], default 1.0. Bigger means faster prepays)
- double TuneCATO

  Scale CATO Effect.
- double TuneCashout

  Scale Cashout SMM(range [0,2], default 1.0. Bigger means faster prepays)
- double TuneCure

  Scale Cure SMM(range [0,2], default 1.0. Bigger means faster prepays)
- double TuneLag

  Adjust Lag(range [0,2], default 1.0. Bigger means more lag)
- double TuneRefi

  Scale Refi SMM(range [0,2], default 1.0. Bigger means faster prepays)
- double TuneSATO

  Scale SATO Effect.
- double TuneScale

  Scale overall SMM(range [0,2], default 1.0. Bigger means faster prepays)
- double TuneSlide

  Slide S-Curve by adding X bp to the spread over CCY (Default 0.0, positive means slower speeds)
- double TuneTurnOver

  Scale Turnover SMM(range [0,2], default 1.0. Bigger means faster prepays)
- int TuningStartYear

  Start year to apply tuning parameters.
- int TuningStartMonth

  Start month to apply tuning parameters.
- int TuningRampMonths

  Number of month for tunings to ramp up to full effect from the start date.
- int TuningEndYear

  End year to apply tuning parameters.
- int TuningEndMonth

  End month to apply tuning parameters.
- int TuningFadeMonths

  Number of months for tunings to fade back to default effect after the end date.

The documentation for this struct was generated from the following file:

- include/WSAAdcoProviderApi.h

21.56 MarkitAdcoScenarioParams Struct Reference

The Scenario parameters users can change from run to run, see ResetADCOScenario().

```
#include <WSAAdcoProviderApi.h>
```

Data Fields

- int HPI_vector_size

  The number of elements of HPI_vector.
- double ∗ HPI_vector

  Vector for HPI.
- int reserved

  Reserved for future use.

The documentation for this struct was generated from the following file:

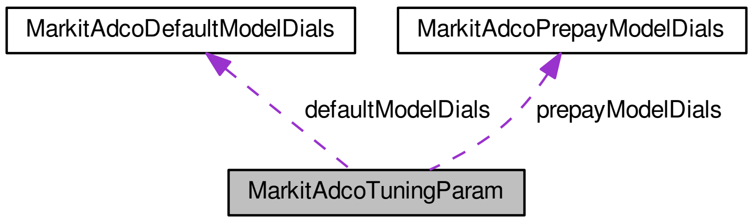
- include/WSAAdcoProviderApi.h

21.57 MarkitAdcoTuningParam Struct Reference

The main tuning parameters for the ADCO default model.

```
#include <WSAAdcoProviderApi.h>
```

Collaboration diagram for MarkitAdcoTuningParam:



Public Types

- enum ERROR_HANDLING { **ON_ERROR_STOP**, **ON_ERROR_CONTINUE** }

  Enum for the the actions when encounter error.
- enum MODEL_TYPES { **MODEL_PREPAY_ONLY**, **MODEL_DEFAULT_AND_PREPAY** }

  Enum for the model types.
- enum DEFAULT_DEFINITIONS { **Repurchase** = 1, **D180**, **Liquidation** }

  Enum for the DEFAULT DEFINITIONS.
- enum CURVE_TYPES { **PAR_SWAP**, **PAR_TSY**, **SPOT_SWAP**, **SPOT_TSY** }

  Enum for the types of curve.
- enum { **DATA_PATH_SIZE** = 1024 }
- typedef int(∗ PROGRESS_CB)(void ∗userData, int pool_processed, int pools_succeeded, int pools_in_deal, char ∗error_message, int max_size_of_error_message)

  The signature of the call back function after process each loan.
- typedef void(∗ LOAN_TUNNING_CB)(void ∗pool_info, void ∗tid, void ∗userData)

  The signature of the call back function before process each loan.

Data Fields

- char AdcoDataPath [DATA_PATH_SIZE]

  ADCo model data path.
- MODEL_TYPES modelType

  ADCo model type: MODEL_PREPAY_ONLY, MODEL_DEFAULT_AND_PREPAY.
- double prepay_magnitude

  Magnitude for prepay rate.
- double default_magnitude

  Magnitude for default rate.
- int recovery_lag

  Recovery lag applyed to each loan.
- int servicer_advancing

  Servicer_advancing;.
- ERROR_HANDLING errorHandling

  Choose what to do when error occurs: stop if set ON_ERROR_STOP, continue if set ON_ERROR_CONTINUE.
- double smmForFailedLoans

  Failed loan's smm in percentage.
- double mdrForFailedLoans

  Failed loan's mdr in percentage.
- double recoveryForFailedLoans

  Failed loan's recovery in percentage.
- int ficoToUseIfNotAvailable

  Fico to use for loans which do not have this INFO.
- int userProvidedPerformanceInfo
- CURVE_TYPES curveType

  Loan is considered subprime if FICO is less than this value (620 if not provided)
- int HPI_vector_size

  The number of elements of HPI_vector.
- double ∗ HPI_vector

  Vector for HPI.
- int ficoSubprimeThreashold
- PROGRESS_CB progressCb

  Called after each loan is processed. If user code installs the cb function, it is up to the user to decide what to do. If somehow the cb finds there are issues, it can return negative code so current api will exit with error.
- void ∗ userData

  When this pointer is set, CB functions will be called with userData set. It is up to user to decide. Set to NULL if you do not need it.
- MarkitAdcoPrepayModelDials ∗ prepayModelDials

  Fine tune params, set to 0 if not needed.
- MarkitAdcoDefaultModelDials ∗ defaultModelDials

  Fine tune params, set to 0 if not needed.
- int logLevel

  Log level for write_log(). For CHS deal only for now. level: 0, no log, 1, message box for windows, >=2 and log file open, log to log file (see create_deal_scenario_object()).
- double recovery_magnitude

  Recovery magnitude.
- short default_definitions
- bool UseADCO2yr10yrFcst
- LOAN_TUNNING_CB loanTunningCb

  Called before each loan is processed. It is for both. If it is installed, client code can set tuning parameters for each loan processed.

21.57.1 Field Documentation

21.57.1.1 short MarkitAdcoTuningParam::default_definitions

default definitions

Value	Meaning
0	users do not specify; ADCO model would decide the value
MarkitAdcoTuningParam::DEFAULT_DEFINITIONS	users specify default definition as one of the enum

21.57.1.2 int MarkitAdcoTuningParam::ficoSubprimeThreashold

Loan is considered subprime if FICO is less than this value (620 if not provided)

21.57.1.3 bool MarkitAdcoTuningParam::UseADCO2yr10yrFcst

flag of use adco 2yr and 10yr forecast or not.

Value	Meaning
false	(by default),users should set LIBOR 24M and 120M rate vectors.
true	ADCO LDM model uses "Fcst_2yr.txt" and "fcst_10yr.txt" from monthly data files as default values; ADCO Prepayment Model uses "libor2yr_new.txt" and "libor10yr_new" from monthly data files as default values.

21.57.1.4 int MarkitAdcoTuningParam::userProvidedPerformanceInfo

Set this flag to 1 if you replaced collateral, and new collateral has loan performance info.

- If user code has called api replace_collateral(), this flag can be set so user provided loan performance info will be populated to ADCo model. Loan performance info include: "Orig_Face", "State", "Original_LTV", "Credit_Score", "Property_Type", "NumUnits", "Loan_Purpose", "Occupancy", "Documentation", "CanNeg-Am", "ServicingFee", "Cur_Face", "Cur_LTV", "ZipCode", "CurFICO", "IsSecondMortgage", "PrepayPenalty-Percent", and more for ADCo default model, "DelinquencyStatus", "LienPosition", "OrigCombinedLTV", "Cur-TotalLTV", "RecastPeriod", "MaxNegAm", "CurMinimumPayment", "PayCap", "PayResetFreq", "OrigMIFlag", "CurMIFlag", "MIAmount". If loan level data are available (not repline), those info will aslo be populated.

The documentation for this struct was generated from the following file:

- include/WSAAdcoProviderApi.h

21.58 MarkitAftDefaultModelDials Struct Reference

The fine tune parameters for the AFT default model. This supplements structure MarkitAftTuningParam, and is optional.

```
#include <WSAAftProviderApi.h>
```

Data Fields

- double dTransitionToPop00Multiplier

  Multiplier for transition to delinq 0.
- double dTransitionToPop30Multiplier

  Multiplier for transition to delinq 30.
- double dTransitionToPop60Multiplier

  Multiplier for transition to delinq 60.

- double dTransitionToPop90Multiplier

  Multiplier for transition to delinq 90.
- double dTransitionToPopFcMultiplier

  Multiplier for transition to delinq Foreclosure.
- double dTransitionToPopDfMultiplier

  Multiplier for transition to delinq Default.
- double dPrepayFromCurrMultiplier

  Multiplier for Prepayment from current.
- double dPrepayFromPop00Multiplier

  Multiplier for Prepayment from delinq 0.
- double dPrepayFromPop30Multiplier

  Multiplier for Prepayment from delinq 30.
- double dPrepayFromPop60Multiplier

  Multiplier for Prepayment from delinq 60.
- double dPrepayFromPop90Multiplier

  Multiplier for Prepayment from delinq 90.
- double dPrepayFromPopFcMultiplier

  Multiplier for Prepayment from delinq Foreclosure.
- double dDefaultAgeMultiplier

  Multiplier for Default Age.
- double dPaymentIncreaseMultiplier

  Multiplier for payment increase.
- double dAdjustedCurrLTVMultiplier

  Multiplier for Adjusted Current LTV.
- int nApplyDialsToBothFixedOrArmFlag
- double dDefaultRateMultiplier

  Multiplier for Default Rate.
- double dLossSeverityMultiplier

  Multiplier for Loss Severity.

21.58.1 Field Documentation

21.58.1.1 int MarkitAftDefaultModelDials::nApplyDialsToBothFixedOrArmFlag

This flag takes the following values

- -1 If dials are for mapped model type (from defdials.def)
- 0 Apply these dials to both portions of a hybrid
- 1 Apply these dials to only the fixed portion of a hybrid
- 2 Apply these dials to only the arm portion of a hybrid

The documentation for this struct was generated from the following file:

- include/WSAAftProviderApi.h

21.59 MarkitAftPrepayModelDials Struct Reference

The fine tune parameters for the AFT prepay model. This supplements structure MarkitAftTuningParam, and is optional.

```
#include <WSAAftProviderApi.h>
```

Data Fields

- double ∗ projGrossWACPercent

  The AFT prepayment model allows users to pass an array of projected variables WACs (as percents) to be used with fixed rate mortgages. This feature is optional; set this item to zero to disable the feature.
- double rfMultiplier

  A factor to be applied to the refinancing component of the prepay speed for each monthly projection.
- double htMultiplier

  A factor to be applied to the housing turnover component of the prepay speed for each monthly projection.
- double ageMultiplier

  A factor used to extend or shorten the aging effect. Values greater than one extend the aging function. Values less than one shorten the function.
- double burnMultiplier1

  A factor used to increase or diminish the effects of pool burnout. Values greater than one increase the effect; values less than one reduce the effect.
- double burnMultiplier2

  Additional factor used to increase or diminish the effects of pool burnout. Values greater than one increase the effect; values less than one reduce the effect.
- double premiumOriginationAdjFactor

  A factor used to increase or diminish the impact of premium origination on the refinancing component of the prepayment model. The default value is one; setting this member to zero turns off the premium origination effect completely. May take any value from zero to (and including) one.
- int ageShiftInHousingTurnoverAgingFunction

  An additive factor used to adjust the span of the housing aging function. The default is zero; units are months.For example, a value of five results in passing "current age + 5 months" to the housing turnover module.
- int useMultiplicativeOrAdditivePopShiftFlag

  Specifies the type of population shift input to be used. The population shift methods - additive and multiplicative - are mutually exclusive. The default is multiplicative. Select additive by setting this member to one.
- int applyPopShiftFromWhatDateFlag
- double additivePop1ToPop2Shift

  Specifies the magnitude of the additive shift from Population 1 to Population 2.
- double additivePop2ToPop3Shift

  Specifies the magnitude of the additive shift from Population 2 to Population 3. This member functions the same as the prior item except that the shift occurs between populations two and three. (See additivePop1ToPop2Shift above.)
- double curPop1ToPop2Shift

  Specifies the percentage shift from Population 1 to Population 2. Expressed as a value greater than one. For instance, a value of 1.1 specifies a 10% shift from population #1 to population #2.
- double curPop2ToPop3Shift

  Specifies the percentage shift from Population 2 to Population 3. This member functions the same as the prior item except that the shift occurs between populations two and three (See curPop1ToPop2Shift above.)
- int calcHousingSalesStartingFromMRatedateFlag

  Designates to the prepay model when to begin calculating housing sales. Typically, the prepay model will use both historical housing sales figures and consensus economic projections provided by AFT. Set this element to zero to implement that. Set the member value to a non-zero value to direct the model to start calculating housing sales numbers from mrateDateYyyyMm a member of the EspPrepayProjStruct structure. In doing so, historical data and consensus projections are ignored.
- int applyPremiumOriginationAdjFactorOnHousingTurnoverFlag

  Designates to the prepay model whether to apply premium origination factor to housing turnover. Set to one to apply the factor; set to zero to disable the feature.
- int turnOffShortTermMultiplicativeAdjustments

  Determines whether short term adjustment processing is applied. Default value is zero which enables sort term adjustment processing. Set to one to disable the application of short term adjustments.
- int applyPrepaymentMultipliersToFixedPeriodOnlyForHybridARM

Designates how prepay multipliers will be applied to Hybrid ARMs. By default, in the case of Hybrid ARMs prepay multipliers are applied to both the fixed and adjustable rate periods. Set this member to one to direct the prepay model to apply the prepay multipliers in this structure to the fixed period only.

- double elbowShiftForRefiMortgageRateInPercent

  Elbow shift for refinancing mortgage rate (in percent). Default value is zero, i.e. no shift.
- double publicityMultiplierChange

  Additive adjustment to the publicity multiplier. For example, for a publicity multiplier of N, applying this member value (d) is equal to N + d.
- double refiLagInMonth

  Lag shift in months for the refi component of the prepay model. The value is applied to the calculation of the effective mortgage rate for determination of the refinancing incentive. You may enter positive or negatives values including fractions. The default value is zero.
- double changeOfCreditRelatedPrepaymentMultiplier

  Factor used to increase or decrease the credit-related prepayment multiplier. The credit related prepayment is modified by 1 + (value of this member).

21.59.1 Field Documentation

21.59.1.1 int MarkitAftPrepayModelDials::applyPopShiftFromWhatDateFlag

Designates when population shifts will begin being applied; applies regardless of the population shift method specified (additive | multiplicative).

- Set to one to specify that shifts should be applied beginning at the mortgage origination.
- Set to two to specify that shifts should be applied beginning at the settlement date.
- For any other value the prepay model applies the shifts beginning from the first date of the mortgage rate projections. (See mrateDateYyyyMm.)

The documentation for this struct was generated from the following file:

- include/WSAAftProviderApi.h

21.60 MarkitAftScenarioParams Struct Reference

The Scenario parameters users can change from run to run, see ResetAFTScenario().

```
#include <WSAAftProviderApi.h>
```

Data Fields

- int HPI_vector_size

  The number of elements of HPI_vector.
- double ∗ HPI_vector

  Vector for HPI.
- int reserved

  Reserved for future use.

The documentation for this struct was generated from the following file:

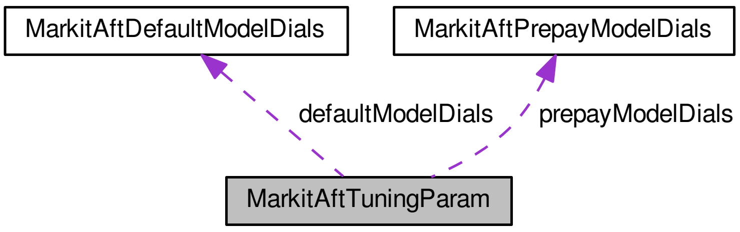
- include/WSAAftProviderApi.h

21.61 MarkitAftTuningParam Struct Reference

The main tuning parameters for the AFT default model.

`#include <WSAAftProviderApi.h>`

Collaboration diagram for MarkitAftTuningParam:



Public Types

- enum APPLY_SPREAD_TO { **APPLY_SPREAD_TO_LIBOR**, **APPLY_SPREAD_TO_TRS** }

  Index to apply spread to.
- enum ERROR_HANDLING { **ON_ERROR_STOP**, **ON_ERROR_CONTINUE** }

  Enum for the the actions when encounter error.
- enum MODEL_TYPES { **MODEL_PREPAY_ONLY**, **MODEL_DEFAULT_AND_PREPAY** }

  Enum for the model types.
- enum { **DATA_PATH_SIZE** = 1024 }
- typedef int(∗ PROGRESS_CB)(void ∗userData, int pools_processed, int pools_succeeded, int pools_in_-deal, char ∗error_message, int max_size_of_error_message)

  The signature of the call back function after process each loan.
- typedef void(∗ LOAN_TUNNING_CB)(void ∗pool_info, void ∗tid, void ∗userData)

  The signature of the call back function before process each loan.

Data Fields

- char AftDataPath [DATA_PATH_SIZE]

  Path to AFT's data directory.
- MODEL_TYPES modelType

  Either AFT prepay model only, or with default.
- double prepay_magnitude

  multiplier to apply to SMMs returned by the model.
- double default_magnitude

  Magnitude for default rate.
- int recovery_lag

  Recovery lag applied to each loan.
- int servicer_advancing

  Servicer advancing.
- double spread_30Y_Mtg_to_30Y_Index

Spread from 30 years mortgage to 30Y index.
- double spread_15Y_Mtg_to_10Y_Index

  Spread from 15 years mortgage to 10Y index.
- double spread_7Y_Mtg_to_7Y_Index

  Spread from 7 years mortgage to 7Y index.
- double spread_5Y_Mtg_to_5Y_Index

  Spread from 5 years mortgage to 5Y index.
- APPLY_SPREAD_TO applySpreadTo

  Which index should apply the spread to.
- ERROR_HANDLING errorHandling

  User can choose to continue or stop at errors.
- double smmForFailedLoans

  SMM in percentage.
- double mdrForFailedLoans

  MDR in percentage.
- double recoveryForFailedLoans

  Recovery in percentage.
- double ltvToUseIfNotAvailable

  Default LTV, in percentage.
- int ficoToUseIfNotAvailable

  Fico to use for loans which do not have this INFO.
- int zipCodeToUseIfNotAvailable

  Default MSA code if not available from input data.
- int userProvidedPerformanceInfo

  Set this flag to 1 if you replaced collateral, and new collareral has loan performance info.
- int HPI_vector_size

  The number of elements of HPI_vector.
- double ∗ HPI_vector

  Vector for HPI.
- int ficoSubprimeThreashold

  Loan is considered subprime if FICO is less than this value (620 if not provided).
- void ∗ userData
- PROGRESS_CB progressCb

  Called after each loan is processed.
- MarkitAftDefaultModelDials ∗ defaultModelDials

  Fine tune params, set to 0 if not needed.
- int checkResultsFor_NAN_Values

  Set to non-0 to verify the NAN values for each period, or 0 to disable the check. The default value is 0.
- MarkitAftPrepayModelDials ∗ prepayModelDials

  Fine tune params, set to 0 if not needed.
- LOAN_TUNNING_CB loanTunningCb

  Called before each loan is processed.

21.61.1 Field Documentation

21.61.1.1 void∗ MarkitAftTuningParam::userData

When this pointer is set, CB functions will be called with userData set. Set to NULL if you do not need it.

The documentation for this struct was generated from the following file:

- include/WSAAftProviderApi.h

21.62 METRIC_INPUT_STRUCT Struct Reference

This structure stores inputs information for metrics calculation.

```
#include <wsa.h>
```

Data Fields

- double clean_price

  Given clean price which the metrics analysts base on.
- APPLY_SPREAD_TYPE apply_spread_to

  Apply spread to Libor curves or Treasury curves, refer to enum APPLY_SPREAD_TYPE.
- bool calc_basic_metrics_only

  If true, just calculate and return DV01, DV100 and CS01 only.

21.62.1 Detailed Description

See Also

- get_bond_market_risk_metrics()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.63 METRIC_INPUT_STRUCT_EX Struct Reference

This structure stores inputs information for metrics ex calculation.

```
#include <wsa.h>
```

Data Fields

- OAS_CAL_MODE oas_mode

  Flag to enable OAS-related metrics calculation or not.
- double shift_amt

  Rate shift amount, a decimal, whose default is 0.00001 (1bp).
- int num_paths

  Number of simulation paths for OAS.
- short shift_index_array [MAX_INDEX_TYPES]

  Array of index rates to shift. Please refer to INDEX_TYPE for available index types, reserve for future use.
- int enable_parallel_run

  Specify number of threads to run OAS simultaneously. Set 0 or 1 would disable parallel run OAS simultaneously.
- bool enable_CMM_custom_scenario_for_CMBS

  Enable CMM custom scenario analysis for each interest rate path for OAS analysis. Credit model must also be set to CMM using set_moodys_credit_model_settings().
- bool disable_fixed_assumption

  Disable the fixed credit assumption for bankloan's risk_metrics_ex calculation.
- bool enable_price_array

  Enable calculating price_array or not.

21.63.1 Detailed Description

See Also

- get_bond_market_risk_metrics_ex()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.64 METRIC_RESULTS_STRUCT Struct Reference

This structure stores outputs results for metrics calculation.

```
#include <wsa.h>
```

Data Fields

- double i_spread

  Interpolated spread, the difference in basis point between its yield to maturity and the linearly interpolated yield for the same maturity on a user-specified reference curve (LIBOR/Treasury).
- double z_spread

  Zero-volatility spread in basis point that makes the security's price equal the present value of its cash flows along each point along the Treasury/Libor curve.
- double macaulay_duration

  The weighted average years to maturity of the cash flows from a security.
- double DV01

  The "dollar value of a 01" - i.e., the estimated change in the price of the instrument given a 1 bp change in the yield.
- double yield_to_worst

  The lowest possible yield on a security.
- double effective_yield

  The yield of a security which has its payments reinvested after they have been received.
- double spread_convexity

  A measure (second derivative) of price sensitivity calculated by shifting z-spread +/- one BP but not the index and measuring average price change.
- double yield_value_of_32nd

  The difference between the initial yield and the new yield given a price change of one tick (i.e., 32nd or 1/32).
- double annual_modified_duration

  The standard macaulay duration formula, assuming an annual compounding frequency.
- double annual_duration_to_worst

  Duration to Worst, based upon an annual compounding frequency.
- double annual_yield_to_maturity

  Yield to Maturity, assuming an annual discounting of coupon payments.
- double annual_yield_to_worst

  Yield to Worst, assuming an annual discounting of coupon payments.
- double CS01

  The "credit spread dollar value of a 01" - i.e., the estimated change in the price of the instrument given a 1 bp change in the discount margin.
- double DV100

  The "dollar value of a 01" - i.e., the estimated change in the price of the instrument given a 1 percent change in the yield.

21.64.1 Detailed Description

See Also

- get_bond_market_risk_metrics()

The documentation for this struct was generated from the following file:

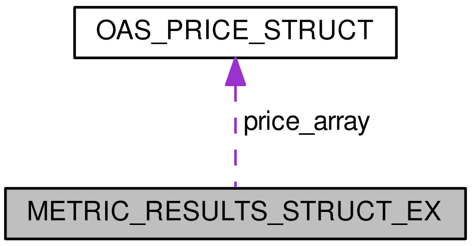
- include/wsa.h

21.65 METRIC_RESULTS_STRUCT_EX Struct Reference

This structure stores metircs results information.

`#include <wsa.h>`

Collaboration diagram for METRIC_RESULTS_STRUCT_EX:



Data Fields

- double OAS

  The constant spread which, when added to the LIBOR/Treasury curve associated with the security's currency, causes the present value of the expected option adjusted cash flows to equal the input price.
- double effective_duration

  The average percentage change in a security's market value (incorporating the OAS) given a user specified shift (e.g., +-100bps) in the corresponding LIBOR/Treasury curve, while incorporating the optionality based on the yield curve shifts. The shift is performed on the spot curve.
- double effective_duration_par

  The average percentage change in a security's market value (incorporating the OAS) given a user specified shift (e.g., +-100bps) in the corresponding government yield curve, while incorporating the optionality based on the yield curve shifts. The shift is performed on the par curve before forward curve construction.
- double effective_convexity

  An option-adjusted measure of the curvature in the relationship between security prices and security yields that demonstrates how the effective duration of a security changes with interest rate shifts (e.g., +/-100 bps).
- double effective_convexity_par

  An option-adjusted measure of the curvature in the relationship between security prices and security yields that demonstrates how the effective duration (par) of a security changes with interest rate shifts (e.g., +/-100 bps).
- double price

  In percentage of a security's current face, the average present value of all the future cash flow paths subtracted by accrued interest, incorporating the optionality based on the simulated economic scenarios. The discount factors are based on the LIBOR/Treasury, OAS and user-specified shift amount.

- double spread_duration

  The average percentage change in a security's market value given user-specified shifts (e.g., +/- 100bps) in its OAS.
- OAS_PRICE_STRUCT price_array

  Optional, only when anchor_type is OAS and METRIC_INPUT_STRUCT_EX.enable_price_array set to true returned. In percentage of a security's current face, the average present value of all the future cash flow paths subtracted by accrued interest, incorporating the optionality based on the simulated economic scenarios. The discount factors are based on the LIBOR/Treasury, OAS and user-specified shift amount and user input settlement date.

21.65.1 Detailed Description

See Also

- get_bond_market_risk_metrics_ex()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.66 MONTE_CARLO_ASSUMPTION Struct Reference

This struct stores settings of monte carlo assumption.

```
#include <wsa.h>
```

Data Fields

- short mode

  simulation mode, 0 for "Auto" and 1 for "Input".
- short num_paths

  number of paths for the simulation to run
- short optimization_type

  options for the simulation to run faster. refer to enum MONTE_CARLO_OPTIMIZATION
- double optimization_pct

  percentage of paths or tail runs for the optimization
- short reinv_pool

  pool used for reinvestment, 0 for "REINV" and 1 for "CURRENT"
- BOOLYAN default_to_reinv

  choose to whether apply default to "REINV" pool.

21.66.1 Detailed Description

New feature Subject to change

See Also

run_monte_carlo_simulation()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.67 MONTE_CARLO_DEF_PPY_REC_ASSUMPTION Struct Reference

This struct stores settings of monte carlo economy assumption.

`#include <wsa.h>`

Data Fields

- short correlation_type

  choose which correlation matrix to use, 0 for "Industry Correlation", 1 for "Portfolio Correlation" and 2 for "Global Correlation(SFW)".
- double intra_industry_correlation

  correlation for assets who are in the same industry.
- double inter_industry_correlation

  correlation for assets whose industries are not in the industry matrix.
- double global_correlation

  global correlation for SFW.
- short default_probability_source

  choose what data source to use for the default probability, refer to enum MONTE_CARLO_DEFAULT_TYPE.
- double default_probability_multiplier

  multiplier for the default probability.
- int ppy_type
- double ppy_pct

  prepayment rate.
- double rec_vol

  volatility for recovery rate.
- double global_rec_correlation

  global recovery correlation.
- double asset_def_rec_correlation

  asset default-recovery correlation.
- short copula

  copula function, 0 for Gaussian and 1 for Student-T.
- short degrees_of_freedom
- bool use_random_seed

  use random seed or not.
- int random_number_seed

  random number seed.

21.67.1 Detailed Description

New feature Subject to change

See Also

run_monte_carlo_simulation()

21.67.2 Field Documentation

21.67.2.1 short MONTE_CARLO_DEF_PPY_REC_ASSUMPTION::degrees_of_freedom

degrees of freedom.

- 0 for degree 3, 1 for degree 4, 2 for degree 5, 3 for degree 6
- 4 for degree 7, 5 for degree 8, 6 for degree 9, 7 for degree 10
- 8 for degree 12, 9 for degree 15, 10 for degree 30, 11 for degree 60

21.67.2.2 int MONTE_CARLO_DEF_PPY_REC_ASSUMPTION::ppy_type

type of prepayment curve.

- PREPAY_CURVE_PSA - Standard prepayment curve measuring for prepayments in the residential mortgage market.
- PREPAY_CURVE_SMM - Monthly prepayment or default rate.
- PREPAY_CURVE_CPR - Constant Prepayment Rate(CPR): Prepayment percentage expressed as an annual compounded rate.
- PREPAY_CURVE_CPY - Constant Prepayment Yield(CPY): It is equivalent to the Constant Prepayment Rate(CPR) except that it assumes prepayment only happens after contractual lockout and yield maintenance period.
- PREPAY_CURVE_HEP - Home Equity Prepayment: A measure of prepayments for closed-end, fixed rate HEL loans. This curve accounts for the faster seasoning ramp for home equity loans.
- PREPAY_CURVE_ABS - Asset-Backed Securities(ABS): It is used in ABS markets, where prepayments differ significantly from standard mortgages. This model defines an increasing sequence of monthly prepayment rates, which correspond to a constant absolute level of loan prepayments in all future periods.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.68 MONTE_CARLO_RESULT Struct Reference

This struct stores results of monte carlo run.

```
#include <wsa.h>
```

Data Fields

- char tranche_name [20]

  name of the tranche
- double total_cashflow

  total cashflow received by the tranche
- double total_principal

  principal received by the tranche
- double breakeven_probability

  break even probability for the tranche
- double tranche_delta

tranche delta
- double irr_average

  average internal rate of return
- double irr_sd

  standard deviation of the internal rate of return
- double price_average

  average price
- double price_sd

  standard deviation of the price
- double yield_average

  average yield
- double yield_sd

  standard deviation of the yield
- double yield_dm_average

  average discounted margin
- double yield_dm_sd

  standard deviation of the discounted margin
- double average_life_average

  average of the average life
- double average_life_sd

  standard deviation of the average life
- double expected_losses_average

  average expected losses
- double expected_losses_capped_average

  average expected capped losses
- double expected_losses_sd

  standard deviation of the expected losses
- double expected_losses_capped_sd

  standard deviation of the expected capped losses
- double **accrued**
- double convexity_average

  average of convexity
- double convexity_sd

  standard deviation of convexity
- char currency [4]
- double duration_average

  average of duration
- double duration_sd

  standard deviation of duration
- double modified_duration_average

  average of modified duration
- double modified_duration_sd

  standard deviation of modified duration
- double pv_at_coupon_average

  average of pv at coupon
- double pv_at_coupon_sd

  standard deviation of pv at coupon
- double reimbursed_loss_average

  average of reimbursed loss
- double reimbursed_loss_sd

  standard deviation of reimbursed loss
- double total_interest

  total interest received by the tranche

21.68.1 Detailed Description

New feature Subject to change

See Also

get_monte_carlo_result()

21.68.2 Field Documentation

21.68.2.1 char MONTE_CARLO_RESULT::currency[4]

currency of tranche Valid Output:

- "USD","GBP","CAD","DEM","UDI",
- "VSM","JPY","CHF","EUR","SDR",
- "ARS","AUD","ATS","BES","BEF",
- "BRL","CLP","CNY","DKK","EGP",
- "FIM","GRD","HKD","ISK","INR",
- "IDR","LUF","MXN","NZD","NOK",
- "PKR","PEN","PHP","RUB","SGD",
- "ZAR","KRW","ESP","SEK","TWD",
- "THB","TRL","UAH","FX1","FX2",
- "DFL","PLN"

The documentation for this struct was generated from the following file:

- include/wsa.h

21.69 MOODYS_ACCOUNT_CASHFLOW Struct Reference

This structure is used to provide user with loan cashflow. It includes number of cashflow points as well as loan cashflow dates.

```
#include <wsa.h>
```

Data Fields

- int size

  Size of the cash flow including element of 0 representing balance at the settlement date.
- int dates [MAX_PERIODS]

  Array of CCYYMMDD dates. The first period stores the last payment date of the current deal update.
- double balance [MAX_PERIODS]

  Balance of account at end of period.
- double withdrawal [MAX_PERIODS]

  Withdrawal amount of account at end of period.
- double deposit [MAX_PERIODS]

  Deposit amount of account at end of period.

- double target [MAX_PERIODS]

  Target amount of account at end of period.
- double interest [MAX_PERIODS]

  Interest amount of account at end of period.
- double deferred_interest [MAX_PERIODS]

  Deferred Interest of account at end of period.
- double commit_fee [MAX_PERIODS]

  Commit_fee of account at end of period.
- double deferred_fee [MAX_PERIODS]

  Deferred_fee of account at end of period.

21.69.1 Detailed Description

See Also

get_loan_flow_ex()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.70 MOODYS_BOND_HISTORY Struct Reference

This structure stores historical information of a given bond.

```
#include <wsa.h>
```

Data Fields

- int month

  Month since origination.
- double ending_balance

  Ending balance.
- double coupon

  Coupon rate.
- double bond_factor

  Bond factor.
- double principal_losses

  Principal losses.
- double cumu_prin_losses

  Cumulative principal losses.
- double paid_interest

  Interest paid out.
- double sub_amount

  Subordination amount.
- double sub_percentage

  Subordination percentage.
- double interest_loss

  Interest shortfall.
- double beginning_balance

  Beginning balance.

- double deferred_interest

  Deferred interest.
- double cumu_interest_losses

  Cumulative interest losses.

21.70.1 Detailed Description

See Also

- get_moodys_bond_history()
- get_moodys_bond_history_avail_YYYYMMs()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.71 MOODYS_BOND_INFO Struct Reference

Additional bond information.

```
#include <wsa.h>
```

Data Fields

- char **tranche_name** [20]
- int vintage_year

  vintage year.
- double subordinate_pct

  percentage of subordination. Available in SFW and CDONET engine.
- double reserve_account_pct

  percentage of reserve account support. Only available in SFW engine.
- char CUSIP [6][10]

  CUSIP.
- char ISIN [6][16]

  ISIN.
- int first_coupon_date

  First payment after origination, format "YYYYMMDD".
- int issue_date

  issue date.
- double issue_price

  issue price.
- char bond_type [20]

  principal type.
- char payment_date_code [11]

  payment date code
- TRANCHE_NULLIFICATION_TYPE nullified

  Tranche nullified value.
- char moodys_bond_id [11]

  Moodys Tranche ID.

21.71.1 Detailed Description

New feature Subject to change

See Also

get_bond_info_by_tranche_ex() get_bond_info_by_index_ex()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.72 MOODYS_DEAL_INFO Struct Reference

Additional deal information.

```
#include <wsa.h>
```

Data Fields

- char full_name [100]

  Full name of deal.
- char country [4]

  Country code of deal.
- int period_begin_date

  Deal period begin date of format YYYYMMDD.
- int period_value

  Period value monthly = 1, quarterly = 3 etc.
- int collateral_update_date

  Collateral update date of format YYYYMMDD. CDOnet engine only.
- char collateral_type [100]

  Collateral type.
- char placement_type [10]

  Placement type.
- char series [15]

  Series.
- char currency [4]

  Currency.
- char product_line [255]

  Product line.
- double effective_date_target_par

  Effective date target par.
- char manager [100]

  Manager for CDOnet.
- char moodys_deal_id [11]

  Moodys deal ID.

21.72.1 Detailed Description

New feature Subject to change

See Also

get_deal_info_ex()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.73 MOODYS_FEE_STRUCT Struct Reference

This structure stores fee information.

```
#include <wsa.h>
```

Data Fields

- int fee_id

  Fee id.
- char fee_name [256]

  Fee name.
- short fee_type

  Fee type calculate, refer to enum MOODYS_FEE_CAL_CODE (SFW only)
- short day_count
- double fee_value

  Value of fee (SFW only)

21.73.1 Detailed Description

See Also

- get_deal_fee()
- set_deal_fee_override()

21.73.2 Field Documentation

21.73.2.1 short MOODYS_FEE_STRUCT::day_count

Day count convention, must be one of: (SFW only)

- DAYCOUNT_DEFAULT
- DAYCOUNT_ACTUAL_360
- DAYCOUNT_ACTUAL_365
- DAYCOUNT_ACTUAL_ACTUAL
- DAYCOUNT_30_360
- DAYCOUNT_30_360E

The documentation for this struct was generated from the following file:

- include/wsa.h

21.74 MOODYS_HEDGE_OVERRIDE Struct Reference

This structure stores hedge override information.

`#include <wsa.h>`

Data Fields

- short counterparty_default_code

  Counterparty default code, 0:no default code, 1: default from.
- int counterparty_default_from

  The start date of counterparty defaults.
- int counterparty_default_to

  The end date of counterparty defaults.
- bool use_paying_margin_override

  Override paying margin or not.
- double override_paying_margin

  Override paying/strike margin, a decimal value.
- int paying_margin_override_from

  The override start date of paying margin.
- int paying_margin_override_to

  The override end date of paying margin.
- bool use_receiving_margin_override

  Override receiving margin or not.
- double override_receiving_margin

  Override receiving margin, a decimal value.
- int receiving_margin_override_from

  The override start date of receiving margin.
- int receiving_margin_override_to

  The override end date of receiving margin.

21.74.1 Detailed Description

See Also

- get_deal_hedge()
- set_deal_hedge_override()

The documentation for this struct was generated from the following file:

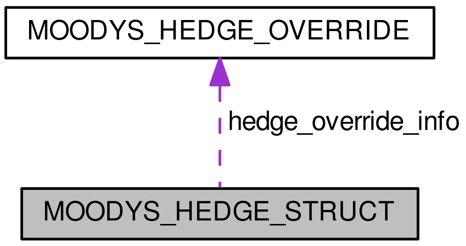
- include/wsa.h

21.75 MOODYS_HEDGE_STRUCT Struct Reference

This structure stores hedge information.

`#include <wsa.h>`

Collaboration diagram for MOODYS_HEDGE_STRUCT:



Data Fields

- char hedge_id [65]

  Hedge name.
- char hedge_desc [256]

  Hedge description.
- char counterparty_name [256]

  Name of counterparty.
- short swap_notional_code

  Swap notional code, refer to enum MOODYS_SWAP_NOTIONAL_CODE .
- short swap_notional_index

  Swap float index, refer to enum INDEX_TYPE and enum INDEX_TYPE_EX.
- double paying_margin

  The paying/Strike margin or the fixed rate that is being applied to the swap, expressed as decimals.
- double receiving_margin

  The receiving margin or the fixed rate that is being applied to the swap, expressed as decimals.
- MOODYS_HEDGE_OVERRIDE hedge_override_info

  The hedge override information.

21.75.1 Detailed Description

See Also

- get_deal_hedge()
- set_deal_hedge_override()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.76 MOODYS_LOAN_CASHFLOW Struct Reference

This structure is used to provide user with loan cashflow. It includes number of cashflow points as well as loan cashflow dates.

```
#include <wsa.h>
```

Data Fields

- int size

  Size of the cash flow including element of 0 representing balance at the settlement date.
- int dates [MAX_PERIODS]

  Array of CCYYMMDD dates. The first period stores the last payment date of the current deal update.
- double balance [MAX_PERIODS]

  Balance at the end of the period.
- double sched_principal [MAX_PERIODS]

  Scheduled principal paid.
- double prepayments [MAX_PERIODS]

  Prepayments per period.
- double defaults [MAX_PERIODS]

  Defaults per period with lag.
- double losses [MAX_PERIODS]

  Losses per period.
- double prin_recoveries [MAX_PERIODS]

  This is principal amount recovered when loan defaults.
- double interest [MAX_PERIODS]

  Interest paid per period.
- double cash [MAX_PERIODS]

  Cash flow per period This field is the sum of interest, scheduled principal, prepayment and principal recoveries from the default loan.
- int start_index_as_per_settle_date
- double draw_amount [MAX_PERIODS]

  Amount of additional money drawn during the draw period, applicable for HELOCs only, (Home Equity Line of Credit). This amount will be added into performing balance in the next period.
- double performing_balance [MAX_PERIODS]

  Performing balance per period.

21.76.1 Detailed Description

See Also

get_loan_flow_ex()

21.76.2 Field Documentation

21.76.2.1 int MOODYS_LOAN_CASHFLOW::start_index_as_per_settle_date

Start index as per settle date

- number of months between trade settlement date (MARKIT_DEAL_INFO::trade_settlement_date) and collateral accrual begin date (MARKIT_BOND_CASHFLOW::accrual_begin_dates[1])

- 0 if trade settlement date is the same or earlier than collateral accrual begin date

The documentation for this struct was generated from the following file:

- include/wsa.h

21.77 MOODYS_POOL_HISTORY Struct Reference

This structure stores historical information of a given pool group.

```
#include <wsa.h>
```

Data Fields

- int month

  Month since origination.
- int loans

  Number of loans in the pool group.
- double ending_balance

  Ending balance.
- double CDR_1M

  Weighted average of the historical 1 month CDR rates.
- double CDR_3M

  Weighted average of the historical 3 months CDR rates.
- double CDR_6M

  Weighted average of the historical 6 months CDR rates.
- double CDR_12M

  Weighted average of the historical 12 months CDR rates.
- double end_del_bal_2M

  Ending delinquency balance 30-59 days.
- double end_del_bal_3M

  Ending delinquency balance 60-89 days.
- double end_del_bal_4M

  Ending delinquency balance 90-119 days.
- double end_del_bal_3M_plus

  Ending delinquency balance 90+ days.
- double serious_del_bal

  Serious delinquencies balance.
- double serious_del_bal_pct

  Serious delinquencies balance percentage.
- double REO_bal

  REO Balance.
- double begin_WAC

  Beginning weighted average coupon.
- double cumu_prin_losses

  Cumulative principal losses.
- double WAM

  Weighted average maturity.
- double CPR_1M

  Weighted average of the historical 1 month CPR rates.
- double CPR_3M

  Weighted average of the historical 3 months CPR rates.
- double CPR_6M

  Weighted average of the historical 6 months CPR rates.
- double CPR_12M

  Weighted average of the historical 12 months CPR rates.
- double chargeoff_severity

Charge off severity.

- double chargeoff_severity_3M

  Charge off severity over 3 months.
- double WALA

  Weighted average loan age.
- double bankruptcy_bal

  Balance in bankruptcy.
- double foreclosed_bal

  Balance in foreclosure.
- double fee_by_servicer

  Service fees rate by servicer.
- double chargeoff_proceeds

  Charge off proceeds.
- int delinq_trigger_breached

  Number of delinquency trigger breached.
- int cumu_delinq_trigger_breached

  Number of cumulative delinquency trigger breached.
- int delinq_cumu_loss_trigger_breached

  Number of cumulative delinquency loss trigger breached.
- double vol_CPR_1M

  Voluntary weighted average of the historical 1 month CPR rates.
- double vol_CPR_3M

  Voluntary weighted average of the historical 3 months CPR rates.
- double vol_CPR_6M

  Voluntary weighted average of the historical 6 months CPR rates.
- double vol_CPR_12M

  Voluntary weighted average of the historical 12 months CPR rates.
- double invol_CPR_1M

  Involuntary weighted average of the historical 1 month CPR rates.
- double invol_CPR_3M

  Involuntary weighted average of the historical 3 months CPR rates.
- double invol_CPR_6M

  Involuntary weighted average of the historical 6 months CPR rates.
- double invol_CPR_12M

  Involuntary weighted average of the historical 12 months CPR rates.
- double CDR_lifetime

  CDR rate since issuance.
- double delinq_trigger_threshold_pct

  Threshold percentage of delinquency trigger.
- double delinq_trigger_curr_level_pct

  Current level percentage of delinquency trigger.
- double cumu_loss_trigger_threshold_pct

  Threshold percentage of cumulative loss trigger.
- double cumu_loss_trigger_curr_level_pct

  Current level percentage of cumulative loss trigger.
- double vol_CPR_lifetime

  Voluntary CPR rate since issuance.
- double invol_CPR_lifetime

  Involuntary CPR rate since issuance.
- double CPR_lifetime

  Reserve for future use.
- double periodic_loss

  Reserve for future use.

21.77.1 Detailed Description

See Also

- get_moodys_pool_history()
- get_moodys_pool_history_avail_YYYYMMs()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.78 MOODYS_POOL_INFO Struct Reference

Data Fields

- int loan_number

  An ordinal for loans inside a deal.
- bool bank_loan

  Whether the loan is bank loan or not in CDOnet.
- bool corporate_bond

  Whether the loan is corporate bond or not in CDOnet.
- char seniority [15]

  Seniority name of a loan as Junior/Senior in CDOnet.
- char currency [4]

  Currency of the loan.
- bool structured_security

  Whether the loan is structured security or not.
- char cusip [10]

  The pool cusip.
- int initial_penalty_months

  The initial penalty months.
- int total_prepay_penalty_months

  The total prepay penalty months.
- PREPAY_PENALTY_STRUCTURE prepay_penalty_structure

  The prepay penalty structure.
- int ever_construction_flag

  The ever construction flag.
- char loanx_id [20]

  Markit's loan unique identifier.
- int moodys_rating

  Moodys Rating.
- int maturity_date

  maturity date
- double market_price_1

  Market Price 1.
- int next_pay_date

  Loan level next pay date.
- char moodys_issuer_name [60]

  Pool's moodys issuer name.
- char global_issuer_name [60]

  Pool's global issuer name.

- char company [255]

  Pool's company name.
- char moodys_category [90]

  Pool's category.
- char country [30]

  Pool's country.
- char asset_group [200]

  Pool's asset group.
- char asset_type [128]

  Pool's asset type.
- char whole_loan_id [101]

  An ordinal for loans inside a whole loan.
- int issue_age

  Loan issue age.
- int origination_date

  Loan level origination date.
- double recovery_rate

  Asset recovery rate for CDOnet.
- char non_performing [4]

  Asset non-performing flag, "YES" or "NO" or "N/A".
- char MKMV_id [7]

  Asset MKMV ID for CDOnet.
- double DSCR

  Asset DSCR for SFW.
- double NOI

  Asset NOI for SFW.
- int issue_date

  Pool's issue date.
- double appraisal_value

  Asset appraisal value for SFW.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.79 MOODYS_SSFA_CALC Struct Reference

Simplified Supervisory Formula Approach (SSFA) calculation.

```
#include <wsa.h>
```

Data Fields

- double Kg
- double W
- double A
- double D
- double Ps
- double Ks
- double Ka
- double Kssfa
- double RW

21.79.1 Detailed Description

New feature Subject to change

See Also

get_moodys_ssfa_calc()

21.79.2 Field Documentation

21.79.2.1 double MOODYS_SSFA_CALC::A

Attachment point of exposure, the threshold at which credit losses will first be allocated to the position. Call get_-bond_flow_ex() with identifier FLOW_BOND_ATTACHMENT to get projections.

- must be a decimal between 0 and 1

21.79.2.2 double MOODYS_SSFA_CALC::D

Detachment point of exposure, the threshold at which credit losses of principal allocated to the position would result in total loss of principal. Call get_bond_flow_ex() with identifier FLOW_BOND_DETACHMENT to get projections.

- must be a decimal between 0 and 1

21.79.2.3 double MOODYS_SSFA_CALC::Ka

Consolidated total capital requirement of the underlying exposures Call get_bond_flow_ex() with identifier FLOW_-BOND_SSFA_KA to get projections.

```
Ka = (1-W)*Kg + 0.5*W
```

21.79.2.4 double MOODYS_SSFA_CALC::Kg

Weighted average (unpaid principal for weighting) total capital requirement of the underlying exposures Call get_-bond_flow_ex() with identifier FLOW_BOND_SSFA_KG to get projections.

- must be a decimal between 0 and 1

21.79.2.5 double MOODYS_SSFA_CALC::Ks

Weighted average capital charge on the underlying structured securities

- must be a decimal between 0 and 1

21.79.2.6 double MOODYS_SSFA_CALC::Kssfa

The SSFA formula used for capital charge calculation (see RW). Call get_bond_flow_ex() with identifier FLOW_B-OND_SSFA_KSSFA to get projections.

```
Kssfa = [e^(a*u)-e^(a*l)]/[a*(u-l)]
```

Where

```
a = -1/(p*Ka)
u = D - Ka
l = max(A - Ka, 0)
```

Where p is the resecuritization factor:

```
p = 0.5 for securitization positions, or for resecuritization positions that have one single asset
p = 1.5 for all other resecuritization positions
```

21.79.2.7 double MOODYS_SSFA_CALC::Ps

Percentage of structured assets in the underlying exposures

- must be a decimal between 0 and 1

21.79.2.8 double MOODYS_SSFA_CALC::RW

Risk weight based on SSFA calculation methodology. Call get_bond_flow_ex() with identifier FLOW_BOND_SSF-A_RW to get projections.

```
If  [Ka] >= [D] then
    [RWCap] '1250%
Elseif  [A] >= [Ka] then
    If [Kssfa_Cap] < [RWFloor] then
        [RWFloor] '20%
    Else
        [Kssfa_Cap]
    End If
Elseif [Opt3] < [RWFloor] Then
    [RWFloor]
Else
    [Opt3]
End If
```

Where

```
[Kssfa_Cap] = Kssfa*RWCap
[Opt3] = ((Ka-A)/(D-A)*RWCap)+((D-Ka)/(D-A)*Kssfa*RWCap)
```

CapCharge, the percentage used to determine the capital charge for the security, is

```
CapCharge = RW/1250%
```

21.79.2.9 double MOODYS_SSFA_CALC::W

Ratio of sum of dollar amounts of any underlying exposures default or 90+ days delinquent over the balance of underlying exposures Call get_bond_flow_ex() with identifier FLOW_BOND_SSFA_W to get projections.

- must be a decimal between 0 and 1

The documentation for this struct was generated from the following file:

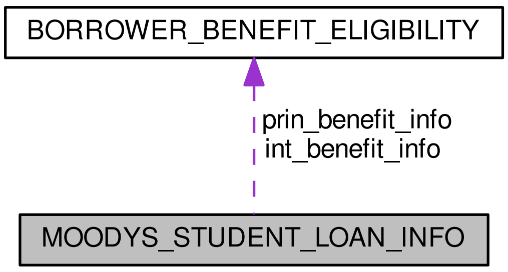
- include/wsa.h

21.80 MOODYS_STUDENT_LOAN_INFO Struct Reference

This structure stores loan level information for a student loan.

```
#include <wsa.h>
```

Collaboration diagram for MOODYS_STUDENT_LOAN_INFO:



Data Fields

- short loan_type
- BOOLYAN subsidized

  Indicates whether interest is subsidized by the government.
- short deferment_code
- short int_cap_period
- double min_payment

  Minimum payment for the loan.
- short input_payment_type
- short input_payment_from

  Indicates when schedule payment begins.
- double input_payment_amount [MAX_PERIODS]

  Payment schedule.
- BOOLYAN borrower_reduct_at_default

  If number of borrower will be reducted when loan defaults.
- double repay_gross_margin

  Gross margin in repayment period.
- double orig_accrued_int

  Original accrued interest.
- double update_accrued_int

  Updated accrued interest.
- short orig_nonpyament_term

  Original non-payment term.
- short update_nonpayment_term

  Updated non-payment term.
- short orig_repay_term

  Original repayment term.
- short update_repay_term

Updated repayment term.

- STUDENT_LOAN_STATE orig_loan_state

  Original loan status.
- short orig_months_in_repay

  Original months in repay.
- short update_months_in_repay

  Updated months in repay.
- short orig_borrower_count

  Original borrower number.
- short update_borrower_count

  Updated borrower number.
- BORROWER_BENEFIT_ELIGIBILITY int_benefit_info [6]

  Borrower interest benefit.
- BORROWER_BENEFIT_ELIGIBILITY prin_benefit_info

  Borrower principal benefit.
- double prin_benefit_applied

  Amount of principal benefit applied.
- BOOLYAN sap_eligible

  Indicates whether the loan is eligible for special allowance payment.
- BOOLYAN sap_in_deferment

  Indicates whether sap is paid in deferment.
- BOOLYAN excess_int_to_doe

  Indicates whether excess interest is paid to DOE (Department of Education)
- short sap_index

  Index used for special allowance payment.
- double defer_sap_margin

  Sap margin in deferment.
- double repay_sap_margin

  Sap margin in repayment.
- BOOLYAN only_pay_sap_for_loans_at_cap

  Indicate if special allowance payment is only paid at cap.

21.80.1 Detailed Description

See Also

view_moodys_student_loan_info()

21.80.2 Field Documentation

21.80.2.1 short MOODYS_STUDENT_LOAN_INFO::deferment_code

Deferment type of the loan:

Value	Meaning
1	defer both interest and principal;
2	defer principal only.

21.80.2.2 short MOODYS_STUDENT_LOAN_INFO::input_payment_type

Schedule payment type:

Value	Meaning
1	the loan has a level payment for principal and interest;
2	the loan has a payment schedule for principal;
3	the loan has payment schedule for both principal and interest

21.80.2.3 short MOODYS_STUDENT_LOAN_INFO::int_cap_period

Indicates the periodicity of interest capitalization:

Value	Meaning
0	monthly
3	quarterly
6	semi-annually
12	annually

21.80.2.4 short MOODYS_STUDENT_LOAN_INFO::loan_type

Type of the loan:

Value	Meaning
0	FFELP
1	Stafford
4	Consolidation
6	PLUS
7	SLS
9	HEAL
10	MedLoans
11	Law Access
12	MBA Access
15	Private
16	Other
19	TERI

The documentation for this struct was generated from the following file:

- include/wsa.h

21.81 OAS_PRICE_STRUCT Struct Reference

This structure stores OAS price array.

```
#include <wsa.h>
```

Data Fields

- int date [MAX_PERIODS]

  Monthly date array starting from user specified trade settlement date.
- double price [MAX_PERIODS]

  The corresponding price array calculated from OAS.

21.81.1 Detailed Description

See Also

- get_bond_market_risk_metrics_ex()

The documentation for this struct was generated from the following file:

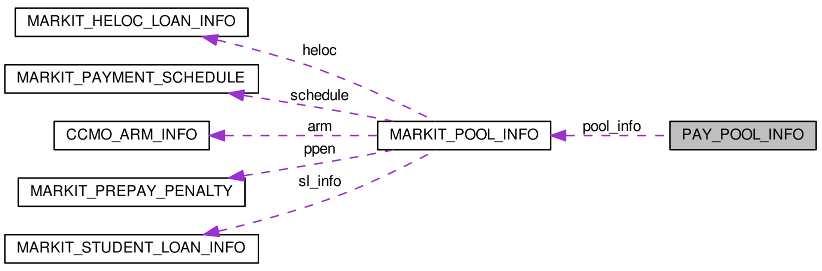
- include/wsa.h

21.82 PAY_POOL_INFO Struct Reference

This structure is passed to assumption call back function.

`#include <ccmo.h>`

Collaboration diagram for PAY_POOL_INFO:



Data Fields

- char ∗ deal_name

  Name of the deal current collateral belongs to.
- char ∗ parent_deal_name

  Name of the parent deal for re-remics.
- char ∗ settle_date

  Settlement date.
- MARKIT_POOL_INFO pool_info

  MARKIT_POOL_INFO structure with parameters of the collateral.
- double ∗ wac

  Array of projected WAC.
- void ∗ unique_pool_id

  Unique identifier of the pool. This field can be used as unique pool identifier by call back function.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.83 PAY_POOL_STATE Struct Reference

This structure is passed to per-period assumption call back function. It contains amortization state of the pool up to current period.

`#include <ccmo.h>`

Data Fields

- double ∗ balance

    Current balance of collateral.
- double ∗ wac

    Current weighted-average coupon of the collateral.
- double ∗ sched_principal

    Scheduled principal.
- double ∗ prepayments

    prepayments
- double ∗ defaults

    Defaults.
- double ∗ losses

    Actual losses after recovery.
- double ∗ prin_recoveries

    Principal recoveries.
- double ∗ interest

    Net interest.
- double ∗ reinvestment

    Reinvestment.
- double ∗ cash

    Sum of principal and interest.
- double ∗ po_balance

    Principal-Only balance.
- double ∗ excess_interest

    Interest paid on the balance at the beginning of the period.
- double ∗ negative_amortization

    The amount by which scheduled principal exceeds actual principal.
- double ∗ beg_bal_interest

    Interest paid on the balance at the beginning of the period.
- double ∗ performing_balance

    Performing balance.
- double ∗ new_defaults

    Defaults happening at a given period.
- double ∗ draw_amount

    Draw amount.
- double ∗ total_excess_losses

    Total excess losses.

21.83.1 Detailed Description

All arrays in the structure have length of MAX_PERIODS. All of them are populated up to current period. The client function should not change their values.

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.84 PPC_STRUCT Struct Reference

Data Fields

- char prepay_type [4]
- double ppc_rates [MAX_PERIODS]

  Vector of prospectus prepayment curve.
- SEASONING_TYPE seasoning

  Type of seasoning.

21.84.1 Field Documentation

21.84.1.1 char PPC_STRUCT::prepay_type[4]

Type of prepayment curve. Must be one of: "PSA": Standard prepayment curve measuring for prepayments in the residential mortgage market. "SMM": Monthly prepayment or default rate. "CPR": Constant Prepayment Rate(-CPR): Prepayment percentage expressed as an annual compounded rate. "HEP": Home Equity Prepayment: A measure of prepayments for closed-end, fixed rate HEL loans. This curve accounts for the faster seasoning ramp for home equity loans. "ABS": Asset-Backed Securities(ABS): It is used in ABS markets, where prepayments differ significantly from standard mortgages. This model defines an increasing sequence of monthly prepayment rates, which correspond to a constant absolute level of loan prepayments in all future periods. "MHP", "ADV": Prepayment method that applies immediate prepayment when its issue coupon is set to a specified percentage. "MON": Prepays the entire performing balance of the loan in the month indicated as the rate (Example: 2 MON has a 100% SMM rate of prepayment or default in month 12 and a 0% rate of prepayment or default during all other months). "PCT": Monthly prepayment rate as a percent of the original balance of the loan. "SDA": Standard default curve measure for defaults in the residential mortgage market. "NIS": Prepayment method for auto leases. "MOD", "RMP"

The documentation for this struct was generated from the following file:

- include/wsa.h

21.85 PRICING_RESULTS Struct Reference

This structure used for holding pricing result, including average life, duration, yield etc.

```
#include <ccmo.h>
```

Data Fields

- double AverageLife

  Average life: in years, calculated from bond projected principal payments, payment periods, and settlement date.
- double AccruedInterest

  Accrued interest: interest accrued from the bond accrual begin date to the settlement date based on bond updated coupon rate and day count (e.g., Act/365), in percentage of the bond updated face value.
- double DiscountMargin

  Discount margin: in percentage. The difference between yield and index rate in each payment period.
- double InterestShortfall

  Total interest shortfall in percentage of bond updated face value.
- double ModifiedDuration

  Modified duration: bond duration adjusted by compounding frequency per year.
- double ModifiedConvexity

  Modified convexity: bond convexity adjusted by compounding frequency per year.
- double Price

When yield is given, price is calculated from bond projected cashflows, payment periods and yield.

- double PrincipalWritedown

  Total principal writedown in percentage of bond updated face value.

- double Yield

  Yield: in percentage. When price is given, yield is calculated from price, bond projected cashflows and payment periods.

- double SpreadDuration

  Spread duration: bond duration which calculation applied coupon index rate plus discount margin as discount factor.

21.85.1 Detailed Description

See Also

price_bond()

The documentation for this struct was generated from the following file:

- include/ccmo.h

21.86 RATE_SHIFT_SETTING Struct Reference

This structure stores inputs to setup ECON rate settings.

```
#include <wsa.h>
```

Data Fields

- SCENARIO_RATE_SHIFT_TYPE alwaysUseScenRateShift

  Need to be one of SCENARIO_RATE_SHIFT_TYPE. By default, it is 'SCENARIO_DEAL'. CDOnet only.

- BOOLYAN rateShiftFromSettle

  Whether apply rate shift from settlement date or not. By default, it is false.

- BOOLYAN shiftRelativeToCurrentRates

  Field reserved.

21.86.1 Detailed Description

See Also

- set_rate_shift_setting()

The documentation for this struct was generated from the following file:

- include/wsa.h

21.87 UK_WHOLE_LOAN_INFO Struct Reference

Data Fields

- short UK_region

  Please refer to enum UK_REGION. By default is IGBR.

- short UK_purpose

loan purpose, only for UK loans, 1:Purchase, 2:Re-mortgage, 3:Equity Release, 4:Renovation, 5:Right to Buy, 6-:Investment Mortgage, 7:Debt Consolidation.

- double fund_growth_rate

  Annual growth rate of life fund or investment account. Required for BULINV mortgage,[0,25].
- double collateral_balance

  The value of collateral fund for BULINV mortgage.
- double monthly_fund_deposit

  Amount of money deposited for BULINV mortgage.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.88 WHOLE_LOAN_EXTEND_INFO Struct Reference

Data Fields

- short occupancy

  Please review WHOLE_LOAN_OCCUPANCY_TYPE enum for types.
- short lien_type

  Line position.
- int borrower_birthday

  The borrower's birthday, format "YYYYMMDD".
- double borrower_income

  The borrower's household income.
- BOOLYAN first_time_buyer

  The borrower is first time buyer or not.
- BOOLYAN structured_security

  Whether the loan is structured security or not.

The documentation for this struct was generated from the following file:

- include/wsa.h

21.89 WHOLE_LOAN_SINK_FUND Struct Reference

Data Fields

- short size

  sink fund schedule size
- int pdate [240]

  sink fund payment date info
- double pprin [240]

  sink fund principal amount

The documentation for this struct was generated from the following file:

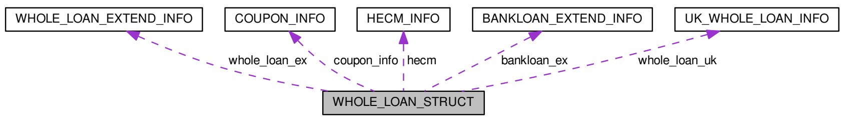
- include/wsa.h

21.90 WHOLE_LOAN_STRUCT Struct Reference

whole loan information.

`#include <wsa.h>`

Collaboration diagram for WHOLE_LOAN_STRUCT:



Data Fields

- char loan_id [10]

  Loan id number.
- short whole_loan_type

  whole_loan_type: type of whole loan; need to be one of "WHOLE_LOAN_TYPE"; regular mortgage cashflow projection would be applied unless being set "WL_HECM" and "WL_REVERSE_MORTGAGE", in which case HECM or Reverse Mortgage methodology would be applied.
- double current_balance

  Current unpaid balance (as of the date the deal is opened).
- double original_balance

  Original balance (as of the date the loan is began).
- short term

  Maturity (in months) of the loan. This should be equal to the number of months between initial date and maturity date.
- short original_term

  The original term in months. This should be equal to the number of months between origination date and maturity date.
- int origination_date

  Origination date, format "YYYYMMDD".
- int first_pay_date

  First payment after origination. format "YYYYMMDD".
- int paid_through

  Paid through, format "YYYYMMDD".
- int maturity

  Loan maturity date, format "YYYYMMDD".
- double ltv

  Loan to value.
- double original_ltv

  Original loan to value.
- double appraisal_value

  Appraisal value.
- short purpose

  Loan purpose.
- char city [25]

  City.
- short state

  State.

- char zip [6]

  Zip.
- char currency [4]

  Standard currency code of the asset,refer to ISO 4217.
- COUPON_INFO coupon_info

  Coupon information, see COUPON_INFO.
- short status

  Loan status, must be one of LOAN_STATUS.
- HECM_INFO hecm

  Hecm information, see HECM_INFO.
- short prin_lockout

  The remaining principal lockout in months (as of the date the deal is opened).
- short balloon_period

  The balloon period.
- char issuer_id [30]

  The issuer ID number.
- char country [4]

  The three-character country codes, please refer to ISO 3166-1 alpha-3.
- short amortization_type

  Loan's amortization method. Need to be one of WHOLE_LOAN_AMORTIZATION_TYPE. By default, it is "ANN".
- short issuer_type

  The issuer type.
- double market_price

  The market price.
- char loan_id_ex [101]

  Loan id number extension only for SFW.
- BOOLYAN individual_pay_date_flag

  Whether enable loan level payment date or not.
- WHOLE_LOAN_EXTEND_INFO whole_loan_ex

  whole loan extend info
- UK_WHOLE_LOAN_INFO whole_loan_uk

  UK whole loan info.
- BANKLOAN_EXTEND_INFO bankloan_ex

  bank loan info, see BANKLOAN_EXTEND_INFO

21.90.1 Detailed Description

New feature Subject to change

See Also

set_whole_loan()

The documentation for this struct was generated from the following file:

- include/wsa.h

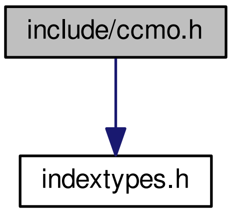
Chapter 22

File Documentation

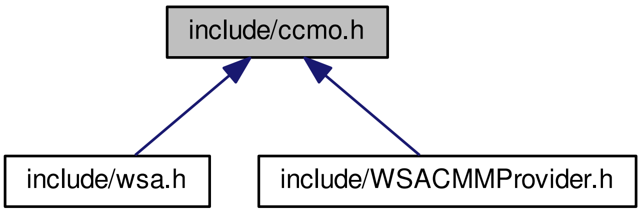
22.1 include/ccmo.h File Reference

```
#include "indextypes.h"
```

Include dependency graph for ccmo.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct CCMO_ARM_INFO

This structure holds adjustable rate information about collateral.
- struct CCMO_POOL_INFO

  This structure holds information about collateral.
- struct CCMO_POOL_INFO_EX

  This structure holds information about collateral.
- struct CCMO_BONDS_S

  This structure holds individual bond information.
- struct MARKIT_BOND_INFO

  This structure holds individual bond information.
- struct CMO_STRUCT

  This is the primary structure used to pass information to and from the WSA API.
- struct MARKIT_PAYMENT_SCHEDULE

  This structure contains two arrays in parallel to describe the payment schedule information.
- struct MARKIT_PREPAY_PENALTY

  This structure contains prepayment penalty information.
- struct MARKIT_HELOC_LOAN_INFO

  This structure contains additional information required by HELOC loans.
- struct MARKIT_STUDENT_LOAN_INFO

  This structure holds information about student loan.
- struct MARKIT_POOL_INFO

  This structure holds information about collateral.
- struct MARKIT_DEAL_INFO

  This structure contains deal level information of a deal.
- struct CMM_CUSTOM_ECONOMIC_DATA
- struct MARKIT_GROUP_INFO

  This structure includes multiple coupon and balance information on a collateral group.
- struct PAY_POOL_INFO

  This structure is passed to assumption call back function.
- struct LICENSE_INFO

  This structure contains license realted information.
- struct PAY_POOL_STATE

  This structure is passed to per-period assumption call back function. It contains amortization state of the pool up to current period.
- struct CCMO_PERIOD_ASSUMPTIONS

  This structure is passed to per-period assumption call back function. Populate the period's element of the arrays with assumptions for the current period. For use of the per period assumptions call back functions, populate the period's element of the arrays to provide assumptions for the current period. All arrays have a size of MAX_PERIODS.
- struct CCMO_COLLAT_ASSUMPTIONS

  This structure is passed to the per-collateral assumptions call back function. Populate its element with assumptions for all amortization periods. For use of the per collateral assumptions call back function, populate assumptions for a given collateral for the entire simulation period.
- struct MARKIT_BOND_CASHFLOW

  This structure is used to provide user with bond cashflow. It has projected cashflow as well as possibly one historical cashflow some 0 delay bonds are entitled to. It includes number of cashflow points as well as specific bond cashflow dates. Previously only projected cashflows were available. However some 0 delay floater bond owners might also be entitled to one historical payment. It will be reported in this structure.
- struct MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE

  This structure is a variation of structure MARKIT_BOND_CASHFLOW.
- struct MARKIT_COLLAT_CASHFLOW

  This structure is used to provide user with collateral cashflow. It includes number of cashflow points as well as collateral cashflow dates.
- struct MARKIT_SURVEILLANCE_DELINQ_DETAILS

  This structure contains prepayment penalty information.

- struct MARKIT_SURVEILLANCE_PERFORM_DETAILS

  This structure contains loan performance, specifically the prepayment speed and default information.
- struct MARKIT_GROUP_SURVEILLANCE_DATA

  This structure is the collateral group level surveillance information for a deal. This structure is contained in deal level structure MARKIT_DEAL_SURVEILLANCE_DATA.
- struct MARKIT_DEAL_SURVEILLANCE_DATA

  This structure is the deal level surveillance information for a deal.
- struct PRICING_RESULTS

  This structure used for holding pricing result, including average life, duration, yield etc.
- struct COLLAT_EXTEND_INFO

  This structure used for COLLAT_ASSUMP_CB_EX to store extended loan info.
- struct MOODYS_POOL_INFO
- struct MARKIT_PAYMENT_BOND

  This structure contains the bond be payment of group.
- struct MARKIT_GROUP_PAYMENT_SET

  This structure contains one set of payment group information.
- struct MARKIT_DEAL_PAYMENT_GROUP

  This structure is the deal level payment group information.
- struct MARKIT_POOL_HISTORY_DATA

  This structure contains the pool history data.

Macros

- #define **VERSION** "4, 0, 1, 0"
- #define **CHAS_LINUX** 1
- #define **CHASAPI**
- #define MAX_PERIODS 612

  The maximum length of the forecasts (including the current period, time 0). The highest index is MAX_PERIODS-1.
- #define MAX_PACS 50

  The maximum number of scheduled balances.
- #define MAX_COLL_GROUPS 25

  The maximum number of collateral groups.
- #define DEFAULT_CURVE_CDR 0

  The Constant Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the year assumed to terminate during the year through events of default.
- #define DEFAULT_CURVE_SDA 1

  The SDA Standard Default Assumptions rate specifies an annual default percentage as a function of the seasoning of the mortgages/loans.
- #define DEFAULT_CURVE_MDR 2

  The Monthly Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the month assumed to terminate during the month through events of default.
- #define DEFAULT_CURVE_SEASONED_CDR 3

  The Constant Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the year assumed to terminate during the year through events of default.
- #define DEFAULT_CURVE_SEASONED_MDR 4

  The Monthly Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the month assumed to terminate during the month through events of default.
- #define **DEFAULT_CURVE_ORIG_MDR** 5
- #define **DEFAULT_CURVE_ORIG_CDR** 6
- #define DEFAULT_CURVE_PCT 7

  The PCT is similar to the MDR curve except that defaults are applied each month to the period 0 balance of the loan, rather than the current balance of the loan.

- #define DEFAULT_CURVE_PLD 8

    The PLD is the Project Loan Default rate curve. This option is available in SFW engine and CHS engine.
- #define PREPAY_CURVE_PSA 1

    Standard prepayment curve measuring for prepayments in the residential mortgage market.
- #define PREPAY_CURVE_SMM 0

    Monthly prepayment or default rate.
- #define PREPAY_CURVE_CPR 2

    Constant Prepayment Rate(CPR): Prepayment percentage expressed as an annual compounded rate.
- #define PREPAY_CURVE_HEP 3

    Home Equity Prepayment: A measure of prepayments for closed-end, fixed rate HEL loans. This curve accounts for the faster seasoning ramp for home equity loans.
- #define PREPAY_CURVE_ABS 4

    Asset-Backed Securities(ABS): It is used in ABS markets, where prepayments differ significantly from standard mortgages. This model defines an increasing sequence of monthly prepayment rates, which correspond to a constant absolute level of loan prepayments in all future periods.
- #define **PREPAY_CURVE_CUS** 5
- #define **PREPAY_CURVE_MHP** 6
- #define **PREPAY_CURVE_SEASONED_SMM** 7
- #define **PREPAY_CURVE_SEASONED_CPR** 8
- #define **PREPAY_CURVE_CPB** 9
- #define PREPAY_CURVE_CPY 10

    Constant Prepayment Yield(CPY): It is equivalent to the Constant Prepayment Rate(CPR) except that it assumes prepayment only happens after contractual lockout and yield maintenance period.
- #define FLOW_BOND_BALANCE 0

    Balance of current period (see EXTENDED_FLOW_BOND_IDENTIFIER for more info)
- #define FLOW_BOND_INTEREST 1

    Interest received in current period.
- #define FLOW_BOND_PRINCIPAL 2

    Principal received by bond holder, including prepayment, default recoveries and schedule principal.
- #define FLOW_BOND_CASH 3

    Sum of interest and principal, prepayment, recoveries from default that received in current period.
- #define FLOW_BOND_PAC_SCHED_REG 4

    Return sinking fund schedule.
- #define FLOW_BOND_PAC_SCHED_MAX 5

    Reserve for future use. Currently it is the same as FLOW_BOND_PAC_SCHED_REG.
- #define FLOW_BOND_PAC_SCHED_MIN 6

    Reserve for future use. Currently it is the same as FLOW_BOND_PAC_SCHED_REG.
- #define FLOW_BOND_CAPPED_RATE 7

    Bond (capped, if any) coupon rate.
- #define CLEANUP_CALL_NONE 0

    No cleanup call.
- #define CLEANUP_CALL_DATE 1

    Callable by date.
- #define CLEANUP_CALL_PERCENT 2

    Callable by percent.
- #define CLEANUP_CALL_EITHER 3

    Callable by date or percent.
- #define CLEANUP_CALL_BOTH 4

    Callable by date and percent.
- #define DAYCOUNT_DEFAULT 0

    Same as DAYCOUNT_30_360.
- #define DAYCOUNT_ACTUAL_360 1

Actual days per month, 360 days per year.

- #define DAYCOUNT_ACTUAL_365 2

  Actual days per month, 365 days per year.
- #define DAYCOUNT_ACTUAL_ACTUAL 3

  Actual days per month, actual days per year.
- #define DAYCOUNT_30_360 4

  30 days per month, 360 days per year
- #define DAYCOUNT_30_365 5

  30 days per month, 365 days per year
- #define DAYCOUNT_30_360E 6

  30 days per month, 360 days per year, last day in Feb = 30th
- #define DAYCOUNT_ACTUAL_AVG 7

  Actual days per month, 365.25 days per year.
- #define DAYCOUNT_SIZE 8

  Number of supported day count conventions.
- #define CUSTOM_AMORT_NONE 0

  WSA API amortizes the collateral.
- #define CUSTOM_AMORT_BY_DEAL 1

  Cashflows are set at the deal level (collateral group 0),SFW engine not supported deal level currently.
- #define CUSTOM_AMORT_BY_GROUP 2

  Cashflows are set by collateral group.
- #define SERVICER_ADVANCES_INTEREST 1

  The servicer advances interest only.
- #define SERVICER_ADVANCES_NOTHING 0

  The servicer advances neither interest nor principal.
- #define SERVICER_ADVANCES_BOTH 2

  The servicer advances both interest and principal.
- #define DEFLT_FROM_CURBAL 0

  See set_default_from_ex() for more info.
- #define DEFLT_FROM_ORIGBAL 1

  See set_default_from_ex() for more info.
- #define DEFLT_FROM_ZERO_CPR 2

  See set_default_from_ex() for more info.
- #define RECOVERY_RATE_AT_RECOVERY 0

  Recovery rate at the time of recovery in case of recovery lag.
- #define RECOVERY_RATE_AT_DEFAULT 1

  Recovery rate at the time of default in case of recovery lag.
- #define LOG_OPTION_SUPPRESS 0

  No console or log file output.
- #define LOG_OPTION_POPUP 1

  Console output only - no log file.
- #define LOG_OPTION_OVERWRITE 2

  Log file only - overwrite the existing file.
- #define LOG_OPTION_APPEND 3

  Log file only - append to the existing file.
- #define LOG_OPTION_OVERWRITE_POPUP 4

  Log file and console - overwrite the existing file.
- #define LOG_OPTION_APPEND_POPUP 5

  Log file and console - append to the existing file.
- #define **DAYTDEF** 1
- #define **BOOLDEF** 1
- #define SBYTE signed char

  Single byte, signed.
- #define **get_chasen_id** get_markit_id

Typedefs

- typedef long DAYT

  SDK date type, expressed as number of days from 1980/01/01.
- typedef short BOOLYAN

  Bool type, 0 for false and non-0 for true.
- typedef void(∗ USER_CLEANUP_CB)(void ∗user_data)
- typedef int(∗ CMM_CUSTOM_ECONOMY_CB)(char ∗input_csv_file, char ∗output_file, CMM_CUSTOM-_ECONOMIC_DATA ∗cmm_economic_data, int cmm_economic_data_length, bool batch_generated, char ∗error_message, int max_size_of_error_message, char ∗custom_scen_name, char ∗zip_extract_path)
- typedef int(∗ COLLAT_ASSUMP_CB)(void ∗tid, char ∗first_period_date, int max_periods, PAY_POOL_IN-FO ∗pool_info, CCMO_COLLAT_ASSUMPTIONS ∗assumptions, void ∗user_data, char ∗error_message, int max_size_of_error_message)
- typedef int(∗ **COLLAT_ASSUMP_CB_EX1**)(void ∗tid, char ∗first_period_date, int max_periods, PAY_P-OOL_INFO ∗pool_info, CCMO_COLLAT_ASSUMPTIONS ∗assumptions, void ∗user_data, char ∗error_-message, int max_size_of_error_message, COLLAT_EXTEND_INFO ∗collat_extend_info, MOODYS_PO-OL_INFO ∗moodys_pool_info)
- typedef int(∗ PER_PERIOD_ASSUMP_CB)(void ∗tid, int period, int max_periods, PAY_POOL_INFO ∗pool-_info, PAY_POOL_STATE ∗pool_state, CCMO_PERIOD_ASSUMPTIONS ∗assumptions, void ∗user_data, char ∗error_message, int max_size_of_error_message)
- typedef void(∗ **AMORT_PROGRESS_CB**)(void ∗tid, int current_pool, int total_pools)
- typedef void(∗ **POOL_CASHFLOW_CB**)(void ∗tid, PAY_POOL_INFO ∗pool_info, int last_populated_period, PAY_POOL_STATE ∗pool_cashflow, CCMO_PERIOD_ASSUMPTIONS ∗assumptions, void ∗user_data)
- typedef USER_CB_RETURN_CODE(∗ INPUT_DIR_CB)(const char ∗deal_name, const char ∗file_name, const char ∗archive_name, const char ∗default_dir_if_noop, char ∗file_dir_to_provide, int size_of_file_dir_-buffer, char ∗error_msg, int size_of_error_msg_buffer)
- typedef USER_CB_RETURN_CODE(∗ **INPUT_DIR_CB_EX**)(const char ∗deal_name, const char ∗file_-name, const char ∗archive_name, const char ∗default_dir_if_noop, char ∗file_dir_to_provide, const char ∗update_date, int size_of_file_dir_buffer, char ∗error_msg, int size_of_error_msg_buffer)

Enumerations

- enum AGENCY_TYPE {
  NON, FNMA_, FNMA_GNMA, FRED_75,
  FRED_GOLD, FRED_GNMA, GNMA_ }
- enum DEAL_CONSTANTS { DEAL_SIZE_OF_DEALNAME =80, DEAL_SIZE_OF_ISSUER =80, DEAL_SI-ZE_OF_UNDERWRITER =20, DEAL_SIZE_OF_ASSET_TYPE =26 }
- enum COLLAT_LEVEL { COLLAT_LEVEL_REPLINES, COLLAT_LEVEL_ACTUAL_LOANS, COLLAT_LE-VEL_USER, COLLAT_LEVEL_ACTUAL_LOANS_LATEST_PARTIAL_UPDATE }
- enum BOND_CF_MODE { BOND_CF_MODE_TRADING, BOND_CF_MODE_HOLDING, BOND_CF_MOD-E_HOLDING_CAPPED_ACCRUAL }
- enum BOND_PAYMENT_DATES_TYPE { BOND_PAYMENT_DATES_THEORETICAL, BOND_PAYMENT-_DATES_BUS_RULES }
- enum CREDIT_CARD_ASSUMP_TYPE {
  CREDIT_CARD_ASSUMP_YIELD, CREDIT_CARD_ASSUMP_REPAYMENT, CREDIT_CARD_ASSUMP_-DEFAULT, CREDIT_CARD_ASSUMP_RECOVERY,
  CREDIT_CARD_ASSUMP_PURCHASE, CREDIT_CARD_ASSUMP_PRINCIPAL_PAYMENT }

  Enum for credit card assumption type.
- enum **SURV_PERFORM_MEASURE_TYPE** { **PERFORM_MEASURE_NOT_SET** =-1, **PERFORM_MEAS-URE_CDR** =0, **PERFORM_MEASURE_CPR** =1, **PERFORM_MEASURE_SIZE** =2 }
- enum **SURV_PERFORM_PERIOD_TYPE** { **PERFORM_PERIOD_NOT_SET** =-1, **PERFORM_PERIOD_1M** =0, **PERFORM_PERIOD_3M** =1, **PERFORM_PERIOD_SIZE** =2 }
- enum **AMORTIZATION_TYPE** { **AMORTIZATION_TYPE_LEVEL_PAYMENT**, **AMORTIZATION_TYPE_S-TRAIGHT_AMORTIZER**, **AMORTIZATION_TYPE_SIZE** }
- enum **ADDITIONAL_LOSS_TYPE** { **BANKRUPTCY_LOSS**, **FRAUD_LOSS**, **HAZARD_LOSS**, **ADDITION-AL_LOSS_TYPE_SIZE** }

- enum **ADDITIONAL_LOSS_UNIT** { **ADDITIONAL_LOSS_RATE_MONTHLY** =0, **ADDITIONAL_LOSS_RA-TE_ANNUAL** =1 }
- enum TMP_DIR_TREATMENT { TMP_DIR_DO_NOTHING, TMP_DIR_REMOVE_THIS_PROCESS, TMP-_DIR_REMOVE_ALL }
- enum PA_MULTIPLIER_TYPE { PA_MULTIPLIER_PREPAY = 0, PA_MULTIPLIER_DEFAULT, PA_MULT-IPLIER_SEVERITY, NUM_PA_MULTIPLIER_TYPE }
- enum EXTENDED_FLOW_BOND_IDENTIFIER {
FLOW_BOND_PRINCIPAL_WRITEDOWN = 50, FLOW_BOND_RATE, FLOW_BOND_FLT_INDEX, FLO-W_BOND_INTEREST_DUE,
FLOW_BOND_INTEREST_SHORTFALL, FLOW_BOND_UNCAPPED_RATE, FLOW_BOND_CAPPED_C-OUPON, FLOW_BOND_BASIS_SHORTFALL,
FLOW_BOND_REIMB_LOSSES, FLOW_BOND_DEFERRED_INTEREST, FLOW_BOND_ATTACHMENT, FLOW_BOND_DETACHMENT,
FLOW_BOND_SSFA_W, FLOW_BOND_SSFA_KA, FLOW_BOND_SSFA_KSSFA, FLOW_BOND_SSFA_-RW,
FLOW_BOND_SSFA_KG, FLOW_BOND_IMPLIED_LOSS, FLOW_BOND_CURR_INS_INT_DUE, FLOW_-BOND_CURR_INS_INT_PAID,
FLOW_BOND_CURR_INS_PRIN_DUE, FLOW_BOND_CURR_INS_PRIN_PAID, FLOW_BOND_PAST_I-NS_INT_DUE, FLOW_BOND_PAST_INS_INT_PAID,
FLOW_BOND_PAST_INS_PRIN_DUE, FLOW_BOND_PAST_INS_PRIN_PAID, FLOW_BOND_IO_BALA-NCE, FLOW_BOND_NONIO_BALANCE,
FLOW_BOND_MONTH_END_ACCRUED_INT }
- enum EXTENDED_FLOW_ACCOUNT_IDENTIFIER {
FLOW_ACCOUNT_BALANCE = 0, FLOW_ACCOUNT_SCHED_WITHDRAWAL = 1, FLOW_ACCOUNT_D-EPOSIT = 2, FLOW_ACCOUNT_TARGET = 3,
FLOW_ACCOUNT_INTEREST = 4, FLOW_ACCOUNT_DEFERRED_INTEREST = 5, FLOW_ACCOUNT_-COMMIT_FEE =6, FLOW_ACCOUNT_DEFERRED_FEE = 7,
FLOW_ACCOUNT_DATES =50 }
- enum EXTENDED_FLOW_COLLATERAL_IDENTIFIER {
FLOW_COLLATERAL_BALANCE = 0, FLOW_COLLATERAL_SCHED_PRINCIPAL = 1, FLOW_COLLATE-RAL_PREPAYMENTS = 2, FLOW_COLLATERAL_DEFAULTS = 3,
FLOW_COLLATERAL_LOSSES = 4, FLOW_COLLATERAL_LIQUIDATIONS = 5, FLOW_COLLATERAL_-PRIN_RECOVERIES = 5, FLOW_COLLATERAL_INTEREST = 6,
FLOW_COLLATERAL_REINVESTMENT = 7, FLOW_COLLATERAL_CASH = 8, FLOW_COLLATERAL_B-OND_VALUE = 9, FLOW_COLLATERAL_PO_BALANCE = 10,
FLOW_COLLATERAL_EXCESS_INTEREST = 11, FLOW_COLLATERAL_DEL_30 = 12, FLOW_COLLAT-ERAL_DEL_60 = 13, FLOW_COLLATERAL_DEL_90P = 14,
FLOW_COLLATERAL_PREPAY_PENALTY = 15, FLOW_COLLATERAL_NEGATIVE_AMORT = 16, FLO-W_COLLATERAL_BEG_BAL_INTEREST = 17, FLOW_COLLATERAL_PO_LOSSES = 18,
FLOW_COLLATERAL_DRAW_AMOUNT = 19, FLOW_COLLATERAL_EXCESS_LOSS = 20, FLOW_COL-LATERAL_STUDENT_LOAN_DELAYED_INTEREST = 21, FLOW_COLLATERAL_AUTO_LEASE_RESID-UE_LOSS = 22,
FLOW_COLLATERAL_AUTO_LEASE_NET_RESIDUE_SCHED_FLOW = 23, FLOW_COLLATERAL_CP-R_1M = 24, FLOW_COLLATERAL_CDR_1M = 25, FLOW_COLLATERAL_END_PERFORMING_BAL = 26,
FLOW_COLLATERAL_DEFAULTS_NO_DELAY = 27, FLOW_COLLATERAL_DELBAL_30 = 28, FLOW_C-OLLATERAL_DELBAL_60 = 29, FLOW_COLLATERAL_DELBAL_90P = 30,
FLOW_COLLATERAL_SCHED_P_AND_I = 50, FLOW_COLLATERAL_PO_SCHED_PRIN = 51, FLOW_C-OLLATERAL_PO_PREPAYMENTS = 52, FLOW_COLLATERAL_PO_PRIN_RECOVERIES = 53,
FLOW_COLLATERAL_PERFORMING_BALANCE = 54, FLOW_COLLATERAL_IO_BALANCE = 55, FLO-W_COLLATERAL_ACCRUED_INTEREST = 56, FLOW_COLLATERAL_ACCURED_INTEREST = 56,
FLOW_COLLATERAL_UNSCHED_PRINCIPAL = 57, FLOW_COLLATERAL_PRINCIPAL = 58, FLOW_C-OLLATERAL_STUDENT_LOAN_BAL_SCH_AND_GRACE = 59, FLOW_COLLATERAL_STUDENT_LOA-N_BAL_DEFERMENT = 60,
FLOW_COLLATERAL_STUDENT_LOAN_BAL_FOREBEARANCE = 61, FLOW_COLLATERAL_STUDEN-T_LOAN_BAL_REPAYMENT = 62, FLOW_COLLATERAL_STUDENT_LOAN_FFELP_INT_RECOVERY = 63, FLOW_COLLATERAL_STUDENT_LOAN_ACCRUED_TO_BE_CAPITALIZED = 64,
FLOW_COLLATERAL_STUDENT_LOAN_CAPITALIZED_INT = 65, FLOW_COLLATERAL_STUDENT_L-OAN_INT_LOSS = 66, FLOW_COLLATERAL_STUDENT_LOAN_MONTHLY_ACCRUAL_SAP_PAYMENT

= 67, FLOW_COLLATERAL_STUDENT_LOAN_SAP_PAYMENT = 68,
FLOW_COLLATERAL_STUDENT_LOAN_MONTHLY_ACCRUAL_SAP_EXCESS_INT_PAID_TO_DOE = 69, FLOW_COLLATERAL_STUDENT_LOAN_SAP_EXCESS_INT_PAID_TO_DOE = 70, FLOW_COLLATERAL_STUDENT_LOAN_MONTHLY_ACCRUAL_SUBSIDY_PAYMENT = 71, FLOW_COLLATERAL_STUDENT_LOAN_SUBSIDY_PAYMENT = 72,
FLOW_COLLATERAL_INTEREST_OF_BUYS = 73, FLOW_COLLATERAL_REINVESTMENT_INCOME = 74, FLOW_COLLATERAL_LOSS_SEVERITY = 75, FLOW_COLLATERAL_BALANCE_OF_BUYS = 76,
FLOW_COLLATERAL_DEFAULT_OF_BUYS = 77, FLOW_COLLATERAL_LOSSES_OF_BUYS = 78, FLOW_COLLATERAL_RECOVERY_OF_BUYS = 79, FLOW_COLLATERAL_SCHED_PRIN_OF_BUYS = 80,
FLOW_COLLATERAL_TOTAL_PRIN_OF_BUYS = 81, FLOW_COLLATERAL_UNSCHED_PRIN_OF_BUYS = 82, FLOW_COLLATERAL_WAC = 1001 }

- enum ERROR_HANDLING_LEVEL { ERROR_HANDLING_LEVEL_LOG_IT, ERROR_HANDLING_LEVEL_STOP_CALCULATION }
- enum DEAL_SEARCH_MODE { **DEAL_SEARCH_FROM_FILE**, **DEAL_SEARCH_FROM_MEMORY** }
- enum CALC_LEVEL { CALC_LEVEL_BASIC, CALC_LEVEL_FULL, CALC_LEVEL_NONE, CALC_LEVEL_FULL_WITH_LOAN }
- enum MISSING_INTEREST_RATES_HANDLING { MISSING_INTEREST_RATES_USE_ZERO, MISSING_INTEREST_RATES_TREAT_AS_ERROR }
- enum PRICING_ANCHORS { PRICE, YIELD, DM }
- enum STUDENT_LOAN_STATE {
STUDENT_LOAN_STATE_REPAY = 0, STUDENT_LOAN_STATE_DEFER, STUDENT_LOAN_STATE_FORBEAR, STUDENT_LOAN_STATE_IN_SCHOOL,
STUDENT_LOAN_STATE_GRACE_PERIOD }

  Enum of the status of student loan.
- enum STUDENT_LOAN_REPAY_TYPE { **STUDENT_LOAN_REPAY_TYPE_REPAY** = 0, STUDENT_LOAN_REPAY_TYPE_FULL_DEFER, STUDENT_LOAN_REPAY_TYPE_PRIN_DEFER }

  Enum of the repay types of student loan.
- enum PREPAY_PENALTY_STRUCTURE {
NOT_AVAILABLE =0, LOCKOUT_ONLY, POINTS_ONLY, YIELDMAINTENANCE_ONLY,
LOCKOUT_TO_POINTS, YIELDMAINTENANCE_TO_POINTS, UNKNOWN_TO_POINTS }

  Enum of the prepay penalty structure.
- enum **STUDENT_LOAN_ASSUM_STATE** { **STUDENT_LOAN_ASSUM_DEFER** = 1, **STUDENT_LOAN_ASSUM_FORBEAR** }
- enum **USER_CB_RETURN_CODE** { **CB_FAIL** = -1, **CB_SUCCESS** =0, **CB_NOOP** =1 }
- enum PA_POOL_VECTOR_TYPE {
PA_DQ_RATE_30, PA_DQ_RATE_60, PA_DQ_RATE_90, PA_CPR,
PA_CDR, PA_SEVERITY, PA_PRINCIPAL_PAYMENT, PA_SELLER_PERCENTAGE,
PA_YIELD, PA_CHARGEOFF, PA_PP_LOC_GROWTH, PA_PP_TEN_GROWTH,
PA_PP_TER_GROWTH, PA_PP_MTN_GROWTH, PA_PP_MTM_GROWTH, PA_REPAYMENT_RATE,
PA_DEFERMENT_RATE, PA_FORBEARANCE_RATE, PA_TOTAL_DQ_RATE, NUM_PA_POOL_VECTOR_TYPES }
- enum MARKIT_POOL_HISTORY {
MARKIT_POOL_HISTORY_CPR1M = 0, MARKIT_POOL_HISTORY_CPR3M, MARKIT_POOL_HISTORY_CPR6M, MARKIT_POOL_HISTORY_CPR12M,
MARKIT_POOL_HISTORY_CPR24M, MARKIT_POOL_HISTORY_CPR_LIFE, MARKIT_POOL_HISTORY_PSA1M, MARKIT_POOL_HISTORY_PSA3M,
MARKIT_POOL_HISTORY_PSA6M, MARKIT_POOL_HISTORY_PSA12M, MARKIT_POOL_HISTORY_PSA24M, MARKIT_POOL_HISTORY_PSA_LIFE,
MARKIT_POOL_HISTORY_DRAW1M, MARKIT_POOL_HISTORY_DRAW3M, MARKIT_POOL_HISTORY_DRAW6M, MARKIT_POOL_HISTORY_DRAW12M,
MARKIT_POOL_HISTORY_DRAW_LIFE, **NUM_OF_MARKIT_POOL_HISTORY** }
- enum CLEAN_UP_CALL_BALANCE_TYPE {
BOND_BAL, END_COLLAT_BAL, BEG_COLLAT_BAL, END_PERFORM_BAL,
BEG_PERFORM_BAL, NUM_CALL_BALANCE_TYPE }
- enum CLEAN_UP_CALL_LINK_TYPE { CLEAN_UP_CALL_LINK_AND, CLEAN_UP_CALL_LINK_OR }
- enum CMM_FACTOR_TYPE { CMM_FACTOR_TYPE_ME, CMM_FACTOR_TYPE_IR, NUM_CMM_FACTOR_TYPE }

- enum CMM_FACTOR {
  CMM_ME_REALGDPGROWTH, CMM_ME_UNEMPRATE, CMM_ME_FEDFUNDSRATE, CMM_ME_TS-Y10Y,
  CMM_ME_CPIINFRATE, CMM_ME_POPGROWTH, CMM_ME_NUMHOUSEHOLDSGROWTH, CMM_ME-_RETAILSALESGROWTH,
  CMM_ME_TOTNONFARMEMPGROWTH, CMM_ME_NOMPERSONALINCGROWTH, CMM_ME_HOME-PRICEGROWTH, CMM_ME_BAACORPYIELD,
  CMM_ME_CREPXIDXGROWTH, CMM_IR_LIBOR1M =20, CMM_IR_LIBOR3M, CMM_IR_LIBOR6M,
  CMM_IR_LIBOR1Y, CMM_IR_BANKPRIME, CMM_IR_DISTRICTCOST, CMM_IR_CD1M,
  CMM_IR_CD3M, CMM_IR_CD6M, CMM_IR_TSY1M, CMM_IR_TSY3M,
  CMM_IR_TSY6M, CMM_IR_TSY1Y, CMM_IR_TSY2Y, CMM_IR_TSY3Y,
  CMM_IR_TSY5Y, CMM_IR_TSY10Y, CMM_IR_TSY30Y, CMM_IR_MMOVINGAVGCMT }

Functions

- long CHASAPI close_deal_ex (void ∗tid, CMO_STRUCT ∗cmos)
- int CHASAPI create_deal_scenario_object (void ∗∗Tid, short ∗LogAction, char ∗LogFile, BOOLYAN ∗Debug)
- const char ∗CHASAPI get_deal_error_msg (void ∗tid)
- void CHASAPI set_deal_error_msg (void ∗tid, const char ∗err)
- long CHASAPI open_deal_ex (void ∗tid, CMO_STRUCT ∗cmos)
- long CHASAPI open_pool_from_file (void ∗tid, const char ∗cusip, const char ∗reserved, int YYYYMMDD_-settlement_date)
- int CHASAPI release_deal_scenario_object (void ∗∗Tid)
- long CHASAPI run_deal_ex (void ∗tid, CMO_STRUCT ∗cmos)
- int CHASAPI set_log_options (void ∗tid, short ∗LogAction, char ∗LogFile, BOOLYAN ∗Debug)
- int CHASAPI set_cleanup_call (void ∗tid, double ∗percentage, CLEAN_UP_CALL_BALANCE_TYPE ∗call_-balance_type, CLEAN_UP_CALL_LINK_TYPE ∗link_type, int ∗yyyymm_date, BOOLYAN set_sup_remic)
- void CHASAPI set_deal_calc_level (void ∗tid, CALC_LEVEL level, int propagate_to_remics)
- CALC_LEVEL CHASAPI get_deal_calc_level (void ∗tid)
- const char ∗CHASAPI get_sdk_build_version ()
- int set_maximum_deal_scenario_objects (int max)
- void CHASAPI set_tmp_dir_treatment (TMP_DIR_TREATMENT tmp_dir_treatment)
- void CHASAPI set_deal_search_mode (DEAL_SEARCH_MODE mode)
- void CHASAPI set_input_path (const char ∗input_path)
- void CHASAPI set_error_handling_level (ERROR_HANDLING_LEVEL level)
- void CHASAPI set_missing_interest_rates_handling (MISSING_INTEREST_RATES_HANDLING handling)
- void CHASAPI enable_same_deal_multithreading (int flag)
- int CHASAPI get_markit_id (const char ∗id, char ∗deal, char ∗bond)
- int CHASAPI get_markit_id1 (const char ∗id, char ∗deal, char ∗bond, char ∗err_buffer, int err_length)
- int CHASAPI get_bond_band (void ∗tid, const char ∗bondid, double ∗pricing_wal, double ∗low, double ∗high)
- void CHASAPI get_bond_by_index_ex (void ∗tid, CCMO_BONDS_S ∗b, long index)
- void CHASAPI get_bond_by_name_ex (void ∗tid, CCMO_BONDS_S ∗b, const char ∗id)
- short CHASAPI get_bond_day_cal_cur_ex (void ∗tid, const char ∗bondid, BOOLYAN use_code, char ∗day-_count, char ∗bus_rules, char ∗currency)
- long CHASAPI get_bond_index_ex (void ∗tid, const char ∗id)
- short CHASAPI get_bond_misc_ex (void ∗tid, const char ∗Bond, BOOLYAN ∗IsSeg, BOOLYAN ∗IsMACR, BOOLYAN ∗IsPO)
- long CHASAPI view_all_bonds_ex (void ∗tid, CCMO_BONDS_S all_bonds[ ])
- void CHASAPI get_pool_by_index_ex (void ∗tid, CCMO_POOL_INFO ∗p, long index)
- CCMO_POOL_INFO ∗CHASAPI get_pool_ptr_by_index_ex (void ∗tid, long index)
- long CHASAPI view_colls_ex (void ∗tid, short index, CCMO_POOL_INFO all_colls[ ], CCMO_POOL_INFO_-EX all_colls_ex[ ], short pool_size, short pool_ex_size, short arm_size)
- void ∗CHASAPI obtain_collat_iterator (void ∗tid, const char ∗reremic_deal_id_or_null)
- MARKIT_POOL_INFO ∗CHASAPI get_next_collat (void ∗tid, void ∗collat_iterator)
- MARKIT_POOL_INFO ∗CHASAPI get_average_collat (void ∗tid, void ∗collat_iterator, int group_number)

- MARKIT_POOL_INFO ∗CHASAPI get_average_collat_by_bond (void ∗tid, void ∗collat_iterator, const char ∗bondid)
- int CHASAPI get_next_collat_for_managed_code (void ∗tid, void ∗collat_iterator, MARKIT_POOL_INFO ∗usr_pool, CCMO_ARM_INFO ∗arm, MARKIT_PAYMENT_SCHEDULE ∗sched, MARKIT_PREPAY_PENALTY prepayPenalty[ ], int sizeOfPpenArray, int ∗hasArm, int ∗hasSched, int ∗hasPpen)
- int CHASAPI get_average_collat_for_managed_code (void ∗tid, void ∗collat_iterator, int group_number, MARKIT_POOL_INFO ∗usr_pool, CCMO_ARM_INFO ∗arm, MARKIT_PAYMENT_SCHEDULE ∗sched, MARKIT_PREPAY_PENALTY prepayPenalty[ ], int sizeOfPpenArray, int ∗hasArm, int ∗hasSched, int ∗hasPpen)
- int CHASAPI view_coll_groups (void ∗tid, int groups_array[ ], int groups_array_length, int ∗total_groups)
- int CHASAPI view_bond_coll_groups (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, int groups_array[ ], int groups_array_length, int ∗total_groups)
- int CHASAPI replace_collateral (void ∗tid, const char ∗reremic_deal_id_or_null, MARKIT_POOL_INFO ∗collat_array[ ], int collat_array_size)
- int CHASAPI get_repline_index_list (void ∗tid, const char ∗reremic_deal_id_or_null, int loan_index, int repline_array[ ], int repline_array_length)
- int CHASAPI get_collateral_id_ex (void ∗tid, const char ∗reremic_deal_id_or_null, int loan_index, const char ∗id_type, char ∗id_array[ ], int id_array_length)
- char ∗CHASAPI dayt_to_str (DAYT julian, char ∗temp)
- char ∗CHASAPI dayt_to_str_with_day_count (DAYT julian, char ∗temp, const int dayCount)
- int CHASAPI deal_has_underlying_deal (void ∗tid)
- int CHASAPI get_cleanup_call_ex (void ∗tid, char ∗CallDate, double ∗CallPct, int ∗CallPctCalc)
- int CHASAPI get_dates_from_upd_ex (void ∗tid, char ∗szArchiveName, int UpdDate[ ])
- long CHASAPI get_deal_info (void ∗tid, const char ∗reremic_deal_id_or_null, MARKIT_DEAL_INFO ∗deal_info)
- long CHASAPI get_group_info (void ∗tid, const char ∗reremic_deal_id_or_null, int group_number, MARKIT_GROUP_INFO ∗group_info)
- int CHASAPI get_deal_issuer_type (void ∗tid, char ∗Issuer, char ∗Type)
- int CHASAPI get_hist_data_ex (void ∗tid, CMO_STRUCT ∗cmos, char ∗bondid, double hist_factor[ ], double hist_coupon[ ])
- int CHASAPI get_hist_data_ex1 (void ∗tid, CMO_STRUCT ∗cmos, char ∗bondid, int date[ ], double principal_losses[ ], double paid_interest[ ])
- const char ∗CHASAPI get_input_path ()
- long CHASAPI get_longest_ex (void ∗tid)
- int CHASAPI get_triggers_avail (void ∗tid, char ∗trigger_names[ ], char ∗trigger_descs[ ])
- short CHASAPI set_custom_amortization_ex (void ∗tid, short newVal)
- short CHASAPI set_collateral_flow_ex (void ∗tid, long group_number, int flow_identifier, short flow_length, double ∗flows, CMO_STRUCT ∗cmo)
- long CHASAPI get_rates_ex (void ∗tid, short ∗ps_rates)
- int CHASAPI get_required_rate_codes (void ∗tid, int ∗rate_codes, int size_of_array_codes)
- double ∗CHASAPI get_rate_ex (void ∗tid, short index)
- long CHASAPI set_rate_ex (void ∗tid, short ∗idx, short vector, double ∗pval)
- long CHASAPI set_prepayments_ex (void ∗tid, short type, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)
- long CHASAPI set_addit_group_delinquencies (void ∗tid, const char ∗reremic_deal_id_or_null, int group_number, short is_vector, int delinq_type, double ∗dqVal)
- int CHASAPI install_per_period_assump_cb (void ∗tid, PER_PERIOD_ASSUMP_CB per_period_assump_cb)
- int CHASAPI install_collat_assump_cb (void ∗tid, COLLAT_ASSUMP_CB collat_assump_cb)
- int CHASAPI install_pool_cashflow_cb (void ∗tid, POOL_CASHFLOW_CB pool_cashflow_cb)
- int CHASAPI set_user_data_for_cb (void ∗tid, void ∗user_data)
- int CHASAPI set_pool_level_user_data_for_cb (void ∗tid, const char ∗reremic_deal_id_or_null, short loan_num, void ∗user_data)
- void ∗CHASAPI get_user_data_for_cb (void ∗tid)
- int CHASAPI install_user_cleanup_cb (void ∗tid, USER_CLEANUP_CB user_cleanup_cb, int invoke_on_deal_close)
- void CHASAPI install_input_dir_callback (INPUT_DIR_CB callback)

- void CHASAPI install_input_dir_callback_ex (INPUT_DIR_CB_EX callback_ex)
- void CHASAPI write_log (void ∗tid, char ∗message, int log_level)
- long CHASAPI is_credit_sensitive_ex (void ∗tid, long loan_num)
- void CHASAPI set_default_from_ex (void ∗tid, const short type, BOOLYAN set_sup_remic)
- long CHASAPI set_defaults_ex (void ∗tid, short type, short is_vector, double ∗pval, long loan_num, BOOLY-AN set_sup_remic)
- long CHASAPI set_recoveries_ex (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_-sup_remic)
- long CHASAPI set_recovery_lag_ex (void ∗tid, short val, long loan_num, BOOLYAN set_sup_remic)
- void CHASAPI set_service_advances_ex (void ∗tid, const int type, BOOLYAN set_sup_remic)
- long CHASAPI clean_up_call_ex (void ∗tid, short state, long loan_num, BOOLYAN set_sup_remic)
- int CHASAPI set_trigger_override (void ∗tid, char ∗trigger_name, short is_vector, SBYTE ∗override)
- int CHASAPI calc_cashflow_offset_ex (void ∗tid, const char ∗bondid, const char ∗settlement_date, int ∗months_offset, int ∗days_accrued, int ∗days_offset)
- double ∗CHASAPI get_bond_flow_ex (void ∗tid, const char ∗bondid, int flow_identifier)
- MARKIT_BOND_CASHFLOW ∗CHASAPI get_bond_flow_ex1 (void ∗tid, const char ∗reremic_deal_id_or_-null, const char ∗bondid)
- int CHASAPI get_bond_flow_ex1_for_managed_code (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE ∗cf)
- int CHASAPI get_bond_info_by_tranche (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, MARKIT_BOND_INFO ∗bond_info)
- int CHASAPI get_bond_info_by_index (void ∗tid, const char ∗reremic_deal_id_or_null, int index, MARKIT_B-OND_INFO ∗bond_info)
- int CHASAPI set_bond_cf_mode (void ∗tid, BOND_CF_MODE mode, BOND_PAYMENT_DATES_TYPE payment_dates_type, int propagate_to_remics)
- int CHASAPI price_bond (void ∗tid, const char ∗bondid, PRICING_ANCHORS anchorType, double anchor-Value, PRICING_RESULTS ∗results)
- short CHASAPI set_bond_flow (void ∗tid, const char ∗bondid, int flow_identifier, short flow_length, double ∗flows)
- char ∗ get_cf_date (int per, char ∗date, void ∗tid)
- char ∗ get_bond_cf_date (int per, char ∗date, void ∗tid, const char ∗bondid)
- double ∗CHASAPI get_collateral_flow_ex (void ∗tid, long group_number, int flow_identifier)
- int CHASAPI get_collateral_flow_ex1 (void ∗tid, int group_number, const char ∗reremic_deal_id_or_null, M-ARKIT_COLLAT_CASHFLOW ∗cf)
- int CHASAPI get_trigger_status (void ∗tid, char ∗trigger_name, SBYTE ∗status)
- int CHASAPI view_reremic_deals (void ∗tid, char ∗dealid, CMO_STRUCT remics[])
- int CHASAPI get_reremic_bond_band (void ∗tid, char ∗dealid, const char ∗bondid, double ∗pricing_wal, dou-ble ∗low, double ∗high)
- int CHASAPI get_reremic_bond_misc (void ∗tid, char ∗dealid, const char ∗Bond, BOOLYAN ∗IsSeg, BOOL-YAN ∗IsMACR, BOOLYAN ∗IsPO)
- CCMO_POOL_INFO ∗CHASAPI get_reremic_pool_ptr_by_index (void ∗tid, char ∗dealid, long index, int &er-ror)
- long CHASAPI view_reremic_colls_ex (void ∗tid, char ∗dealid, short index, CCMO_POOL_INFO all_colls[], CCMO_POOL_INFO_EX all_colls_ex[], short pool_size, short pool_ex_size, short arm_size)
- int CHASAPI get_reremic_triggers_avail (void ∗tid, char ∗dealid, char ∗trigger_names[], char ∗trigger_-descs[])
- long CHASAPI set_reremic_prepayments (void ∗tid, char ∗dealid, short type, short is_vector, double ∗pval, long loan_num)
- long CHASAPI set_reremic_default_from (void ∗tid, char ∗dealid, const short type)
- long CHASAPI set_reremic_defaults (void ∗tid, char ∗dealid, short type, short is_vector, double ∗pval, long loan_num)
- long CHASAPI set_reremic_recoveries (void ∗tid, char ∗dealid, short is_vector, double ∗pval, long loan_num)
- long CHASAPI set_reremic_recovery_lag (void ∗tid, char ∗dealid, short val, long loan_num)
- long CHASAPI set_reremic_service_advances (void ∗tid, char ∗dealid, const int type)
- int CHASAPI set_reremic_trigger_override (void ∗tid, char ∗dealid, char ∗trigger_name, short is_vector, SB-YTE ∗override)

- int CHASAPI get_reremic_trigger_status (void ∗tid, char ∗dealid, char ∗trigger_name, SBYTE ∗status)
- int CHASAPI get_surv_avail_YYYYMMs (void ∗tid, int YYYYMMs[ ], int sizeOfYYYYMMs, int ∗numAvailable)
- int CHASAPI get_surveillance_data (void ∗tid, int YYYYMM, char ∗user_buffer, long size_of_user_buffer, long ∗actual_size)
- int CHASAPI get_deal_surv_data (void ∗tid, MARKIT_DEAL_SURVEILLANCE_DATA ∗survData, int YYYY-MM)
- int CHASAPI replace_collateral_for_managed_code (void ∗tid, const char ∗reremic_deal_id_or_null, MARK-IT_POOL_INFO collat_array[ ], int collat_array_size)
- int CHASAPI get_loan_level_avail_YYYYMMs (void ∗tid, int YYYYMMs[ ], int sizeOfYYYYMMs, int ∗num-Available)
- int CHASAPI get_markit_pool_history_avail_YYYYMMs (const char ∗cusip, int YYYYMMs[ ], int size_YYYY-MMs, int ∗num_available)
- int CHASAPI get_markit_pool_history (const char ∗cusip, const int history_identifier, MARKIT_POOL_HIST-ORY_DATA pool_history[ ], int size_array, int YYYYMM)
- int CHASAPI get_bond_payment_group (void ∗tid, const char ∗bondid, char ∗group_names[ ])
- int CHASAPI get_deal_payment_group (void ∗tid, MARKIT_DEAL_PAYMENT_GROUP group_array[ ], int group_array_size, int ∗num_available)
- int CHASAPI get_markit_bond_pool_history_avail_YYYYMMs (void ∗tid, const char ∗cusip, int YYYYMMs[ ], int size_YYYYMMs, int ∗num_available)
- int CHASAPI get_markit_bond_pool_history (void ∗tid, const char ∗cusip, const int history_identifier, MARK-IT_POOL_HISTORY_DATA pool_history[ ], int size_array, int YYYYMM)
- int CHASAPI get_bond_rate_determination_date (void ∗tid, const char ∗bondid, int ∗determination_date)
- int CHASAPI get_agency_pool_prefix (void ∗tid, char ∗pool_prefix)
- int CHASAPI get_license_info (int num_features, LICENSE_INFO lic_info[ ])

Variables

- const int SUVR_DISTRESS_STATE_DESC_LENGTH = 30

  The maximum length of the description of surveillance distress state.
- const int **SURV_PERFORM_DATA_SIZE** = PERFORM_MEASURE_SIZE ∗ PERFORM_PERIOD_SIZE
- const int SURV_DISTRESS_STATES_NUMBER = 10

  Number of surveillance distress states.

22.1.1 Detailed Description

Version

4.0.1.0

Date

2011-2019

22.1.2 Typedef Documentation

22.1.2.1 typedef int(∗ CMM_CUSTOM_ECONOMY_CB)(char ∗input_csv_file, char ∗output_file, CMM_CUSTOM_ECONOMIC_DATA ∗cmm_economic_data, int cmm_economic_data_length, bool batch_generated, char ∗error_message, int max_size_of_error_message, char ∗custom_scen_name, char ∗zip_extract_path)

The function will be called before run CMM model, allowing the user to provide CMM custom data etc to run the custom CMM scenario. Passing 0 for cmm_custom_cb parameter will remove previously installed function for given tid. After deal is closed functions will also be removed.

Parameters

in	*custom_scen_-name*	The name of cmm custom scenario.
in	*input_csv_file*	CMM custom scenario portfolio data input file.
in	*output_file*	CMM custom scenario output file, if 'high_performance_generated' the output is a zip file, otherwise is a csv file.
in	*cmm_economic-_data*	CMM macro-economic and interest rate data inputs for the custom cmm scenario which user can input via set_cmm_custom_scenario() or set_-macroeconomic_factor_ex.
in	*cmm_economic-_data_length*	It would be the length of the array when me_data is an array, otherwise it should be zero.
in	*batch_generated*	If set to true can batch generate mutiple custom scenarios base on the input array of cmm_economic_data. Custom scenarios files are patched in a zip file. Otherwise it only generat one custom scenario csv file.
out	*error_message*	User can write error message into this buffer.
in	*max_size_of_-error_message*	Maximum size of error_message buffer allocated by the WSA API on user's behalf. If user write a message longer than this field a stack corruption will occur.
in	*custom_scen_-name*	it is optional when use batch generated, otherwise it is need to input.
in	*zip_extract_path*	it is optional, only use when batch generated, if input it would extract batch generated zip file to the zip_extract_path, if not input would not extract zip file.

Returns

- If user returns negative value from the function means the custom cmm callback function failed. Whatever message user had written into error_message buffer will be available by calling get_deal_error_-msg() function.
- If user returns 0 simulation will continue

Note

If batch_generated is true, the cmm_economic_data must be input an array of CMM_CUSTOM_ECONO-MIC_DATA. If the cmm_economic_data input an array, the callback would batch generate mutiple custom scenarios, even batch_generated set to false. If batch generated the custom scenarios, custom_scen_name input must be a an integer number which the custom scenario id from, if not input or invalid input, the custom senario id would from 0.

22.1.2.2 typedef int(∗ COLLAT_ASSUMP_CB)(void ∗tid, char ∗first_period_date, int max_periods, PAY_POOL_INFO ∗pool_info, CCMO_COLLAT_ASSUMPTIONS ∗assumptions, void ∗user_data, char ∗error_message, int max_size_of_error_message)

The function will be called before amortizing every collateral allowing the user to provide prepayments, defaults, recovery,etc. for the entire amortization period. Passing 0 for collat_assump_cb parameter will remove previously installed function for given tid. After deal is closed functions will also be removed.

Parameters

in	*tid*	deal/scenario object identifier which user passed to install_per_period_-assump_cb(), this object can be used for calls to other WSA API functions from inside

in	*first_period_date*	CCYYMMDD date of the first projected period
in	*max_periods*	Maximum number of periods required by simulation
in	*pool_info*	Information about collateral for which amortization is being run
out	*assumptions*	Pointer to the structure user populates to provide assumptions
in	*user_data*	Pointer to the user data , registered through a call to set_user_data_for_cb(). This data can allow user to maintain the state during simulation
out	*error_message*	User can write error message into this buffer. If user returns negative value from call back function simulation will end, run_deal_ex() function will return a negative values and get_deal_error_msg() function will return an error_-message which user specified by population this field
in	*max_size_of_-error_message*	Maximum size of error_message buffer allocated by the WSA API on user's behalf. If user write a message longer than this field a stack corruption will occur

Returns

- If user returns negative value from the function the run_deal_ex() function will stop and return negative value (different from the one returned by user). Whatever message user had written into error_message buffer will be available by calling get_deal_error_msg() function.
- If user returns 0 or any positive value the simulation will continue

22.1.2.3 typedef USER_CB_RETURN_CODE(∗ INPUT_DIR_CB)(const char ∗deal_name, const char ∗file_name, const char ∗archive_name, const char ∗default_dir_if_noop, char ∗file_dir_to_provide, int size_of_file_dir_buffer, char ∗error_msg, int size_of_error_msg_buffer)

The function will be called to set extra input directory before SFW or CHS deal archive files (.SFW, .CHS, .LLD, etc) are accessed if installed by install_input_dir_callback().

Parameters

in	*deal_name*	The name (id) of related deal
in	*file_name*	The file to be accessed
in	*archive_name*	The deal archive to be accessed
in	*default_dir_if_-noop*	The default input path to be used for the deal if the function returns CB_NOOP. It should be the input path set by set_input_path()
out	*file_dir_to_-provide*	User should populate extra input path, if applicable, for this particular deal archive/file with this field.
in	*size_of_file_dir_-buffer*	The size of the extra input path buffer
out	*error_msg*	User can write error message into this buffer. If user returns CB_FAIL from call back function, file open will end. get_deal_error_msg() function will return an error_message populated with this field
in	*size_of_error_-msg_buffer*	The size of error_message buffer allocated by the WSA API on user's behalf. If user write a message longer than this field a stack corruption will occur

Return values

CB_FAIL	Calling function from WSA API will fail. error_msg will be populated.
CB_SUCCESS	WSA API will use file_dir_to_provide populated by user function as the input path for the given deal/archive/file
CB_NOOP	WSA API will use default_dir_if_noop (which is set with set_input_path()) as the input path for the given deal/archive/file

Note

This call back function must be thread-safe if it is used within multi-threaded environments. The call back function also needs to use file locks or other mechanisms if there are concurrent writes to file system among different processes.

22.1.2.4 typedef int(∗ PER_PERIOD_ASSUMP_CB)(void ∗tid, int period, int max_periods, PAY_POOL_INFO ∗pool_info, PAY_POOL_STATE ∗pool_state, CCMO_PERIOD_ASSUMPTIONS ∗assumptions, void ∗user_data, char ∗error_message, int max_size_of_error_message)

The function will be called for every simulation period of every collateral allowing the user to examine the current state of collateral and provide assumptions for the next period. Passing 0 for per_period_assump_cb parameter will remove previously installed function for given tid. After deal is closed functions will also be removed.

Parameters

in	*tid*	deal/scenario object identifier which user passed to install_per_period_-assump_cb(), this object can be used for calls to other WSA API functions from inside
in	*period*	Current period of simulation for which user should provide assumptions
in	*max_periods*	Maximum number of periods required by simulation
in	*pool_info*	Information about collateral for which amortization is being run
in	*pool_state*	Information about the state of the pool up to the current period
out	*assumptions*	User should populate period's element of all the arrays inside this structure to provide assumptions for the current period
in	*user_data*	Pointer to the user data , registered through a call to set_user_data_for_cb(). This data can allow user to maintain the state during simulation
out	*error_message*	User can write error message into this buffer. If user returns negative value from call back function simulation will end, run_deal_ex() function will return a negative values and get_deal_error_msg() function will return an error_-message which user specified by population this field
in	*max_size_of_-error_message*	Maximum size of error_message buffer allocated by the WSA API on user's behalf. If user write a message longer than this field a stack corruption will occur

Returns

- If user returns negative value from the function the run_deal_ex function will stop and return negative value (different from the one returned by user). Whatever message user had written into error_message buffer will be available by calling get_deal_error_msg() function.
- If user returns 0 or any positive value the simulation will continue

22.1.2.5 typedef void(∗ USER_CLEANUP_CB)(void ∗user_data)

The signature of the user clean up call back function

Parameters

in	*user_data*	Pointer to the previously registered user data

Returns

None

22.1.3 Enumeration Type Documentation

22.1.3.1 enum AGENCY_TYPE

Can be used to indicate the agency type of the deal (if it is agency deal).

See Also

MARKIT_DEAL_INFO::agency

Enumerator

NON Not one of the listed agency types. Can be any non-agency or agency type not listed.

FNMA_ FNMA, all others.

FNMA_GNMA FNMA, GNMA.

FRED_75 FHLMC, all others.

FRED_GOLD FHLMC, GOLD.

FRED_GNMA FHLMC, GNMA.

GNMA_ GNMA, all others.

22.1.3.2 enum BOND_CF_MODE

Bond cf mode types

See Also

set_bond_cf_mode

Enumerator

BOND_CF_MODE_TRADING Bond cashflow generated as if the bond has been traded (per MARKIT_DEAL_INFO::trade_settlement_date)

BOND_CF_MODE_HOLDING Bond cashflow generated as if the bond is not traded (holding) (per MARKIT_DEAL_INFO::trade_settlement_date)

BOND_CF_MODE_HOLDING_CAPPED_ACCRUAL Bond cashflow generated as if the bond is not traded (holding), accrual interest is capped. This is only for SFW and CDOnet deals.

22.1.3.3 enum BOND_PAYMENT_DATES_TYPE

Bond payment dates types

See Also

set_bond_cf_mode

Enumerator

BOND_PAYMENT_DATES_THEORETICAL Bond payment dates not adjusted for business day rules.

BOND_PAYMENT_DATES_BUS_RULES Bond payment dates adjusted for business day rules.

22.1.3.4 enum CALC_LEVEL

Enum for the calculation level of API.

See Also

set_deal_calc_level() get_deal_calc_level()

Enumerator

CALC_LEVEL_BASIC Basic Calculation Level.

CALC_LEVEL_FULL Full Calculation Level.

CALC_LEVEL_NONE For PA only for now. When call run_deal_ex() with this flag, only PA vectors will be generated and no cashflow generated.

CALC_LEVEL_FULL_WITH_LOAN Full Calculation Level and also with loan level cashflows, currently SFW only.

22.1.3.5 enum CLEAN_UP_CALL_BALANCE_TYPE

Cleanup call balance types.

See Also

set_cleanup_call()

Enumerator

BOND_BAL Bond balance.

END_COLLAT_BAL Ending collateral balance.

BEG_COLLAT_BAL Beginning collateral balance.

END_PERFORM_BAL Ending performing balance.

BEG_PERFORM_BAL Beginning performing balance.

NUM_CALL_BALANCE_TYPE Number of call balance types.

22.1.3.6 enum CLEAN_UP_CALL_LINK_TYPE

Cleanup call link types.

See Also

set_cleanup_call()

Enumerator

CLEAN_UP_CALL_LINK_AND and

CLEAN_UP_CALL_LINK_OR or

22.1.3.7 enum CMM_FACTOR

Enumerator

CMM_ME_REALGDPGROWTH Real GDP Growth.

CMM_ME_UNEMPRATE Unemployment Rate.

CMM_ME_FEDFUNDSRATE Federal Funds Rate.

CMM_ME_TSY10Y 10 year US Treasury

CMM_ME_CPIINFRATE Consumer Price Index Rate.

CMM_ME_POPGROWTH Population Growth Rate.

CMM_ME_NUMHOUSEHOLDSGROWTH Households Growing in Number.

CMM_ME_RETAILSALESGROWTH US Retail Sales Growth.

CMM_ME_TOTNONFARMEMPGROWTH Total Nonfarming Employment Growth.

CMM_ME_NOMPERSONALINCGROWTH Non Personal gGrowth.

CMM_ME_HOMEPRICEGROWTH Home price growth.

CMM_ME_BAACORPYIELD US BAA Corporate Bond Yield.

CMM_ME_CREPXIDXGROWTH CRE Price Index Growth.

CMM_IR_LIBOR1M 1 Month LIBOR

CMM_IR_LIBOR3M 3 Month LIBOR

CMM_IR_LIBOR6M 6 Month LIBOR

CMM_IR_LIBOR1Y 1 Year LIBOR

CMM_IR_BANKPRIME Prime Bank.

CMM_IR_DISTRICTCOST District cost.

CMM_IR_CD1M 1 Month CD

CMM_IR_CD3M 3 Month CD

CMM_IR_CD6M 6 Month CD

CMM_IR_TSY1M 1 Month US Treasury

CMM_IR_TSY3M 3 Month US Treasury

CMM_IR_TSY6M 6 Month US Treasury

CMM_IR_TSY1Y 1 year US Treasury

CMM_IR_TSY2Y 2 year US Treasury

CMM_IR_TSY3Y 3 year US Treasury

CMM_IR_TSY5Y 5 year US Treasury

CMM_IR_TSY10Y 10 year US Treasury

CMM_IR_TSY30Y 30 year US Treasury

CMM_IR_MMOVINGAVGCMT 12 month moving avg CMT

22.1.3.8 enum CMM_FACTOR_TYPE

Enumerator

CMM_FACTOR_TYPE_ME ME data type.

CMM_FACTOR_TYPE_IR IR data type.

NUM_CMM_FACTOR_TYPE NUM CMM FACTOR type.

22.1.3.9 enum COLLAT_LEVEL

Can be used to indicate the collateral level.

See Also

MARKIT_DEAL_INFO::collat_loaded_level

Enumerator

COLLAT_LEVEL_REPLINES Collateral level: repline.

COLLAT_LEVEL_ACTUAL_LOANS Collateral level: actual loan.

COLLAT_LEVEL_USER Collateral level: user defined (collateral replaced by user input, see replace_-collateral())

COLLAT_LEVEL_ACTUAL_LOANS_LATEST_PARTIAL_UPDATE Collateral level: actual loan, and additional actual loan information from partial deal update for requested update date, if available (CHS only)

22.1.3.10 enum CREDIT_CARD_ASSUMP_TYPE

Enumerator

CREDIT_CARD_ASSUMP_YIELD	Portfolio/Annual Yield.
CREDIT_CARD_ASSUMP_REPAYMENT	Repayment Rate.
CREDIT_CARD_ASSUMP_DEFAULT	Default Rate, only use for CHS engine.
CREDIT_CARD_ASSUMP_RECOVERY	Loss Rate.
CREDIT_CARD_ASSUMP_PURCHASE	Purchase Rate.
CREDIT_CARD_ASSUMP_PRINCIPAL_PAYMENT	Principal Payment Rate.

22.1.3.11 enum DEAL_CONSTANTS

Enumerator

DEAL_SIZE_OF_DEALNAME	Max size of deal name is 80 character.
DEAL_SIZE_OF_ISSUER	Max size of issuer name is 80 character.
DEAL_SIZE_OF_UNDERWRITER	Max size of underwriter is 20 character.
DEAL_SIZE_OF_ASSET_TYPE	Max size of asset type is 26 character.

22.1.3.12 enum DEAL_SEARCH_MODE

Enum for deal search mode.

See Also

set_deal_search_mode()

22.1.3.13 enum ERROR_HANDLING_LEVEL

Enum for error handling level.

See Also

set_error_handling_level()

Enumerator

ERROR_HANDLING_LEVEL_LOG_IT	Log error message to log file when error happens.
ERROR_HANDLING_LEVEL_STOP_CALCULATION	Stop running when error happens.

22.1.3.14 enum EXTENDED_FLOW_ACCOUNT_IDENTIFIER

Account cashflow identifiers

See Also

- get_collateral_flow_ex()

Enumerator

FLOW_ACCOUNT_BALANCE	Balance of account cashflow.
FLOW_ACCOUNT_SCHED_WITHDRAWAL	Withdrawal amount of account cashflow.

FLOW_ACCOUNT_DEPOSIT Deposit amount of account cashflow.

FLOW_ACCOUNT_TARGET Target amount of account cashflow.

FLOW_ACCOUNT_INTEREST Interest amount of account cashflow.

FLOW_ACCOUNT_DEFERRED_INTEREST Deferred Interest of account cashflow.

FLOW_ACCOUNT_COMMIT_FEE Deferred commit_fee of account cashflow.

FLOW_ACCOUNT_DEFERRED_FEE Deferred deferred_fee of account cashflow.

FLOW_ACCOUNT_DATES account cashflow dates.

22.1.3.15 enum EXTENDED_FLOW_BOND_IDENTIFIER

Additional bond cashflow identifiers

EXTENDED_FLOW_BOND_IDENTIFIER is the extension to bond cashflow identifiers in ccmo.h:

Bond cashflow identifiers in ccmo.h	Bond cashflow
FLOW_BOND_BALANCE	Balance of current period
FLOW_BOND_INTEREST	Interest received in current period
FLOW_BOND_PRINCIPAL	Principal received by bond holder, including prepayment, default recoveries and schedule principal
FLOW_BOND_CASH	Sum of interest and principal, prepayment, recoveries from default that received in current period
FLOW_BOND_PAC_SCHED_REG	Return sinking fund schedule
FLOW_BOND_PAC_SCHED_MAX	Reserve for future use. Currently it is the same as FLOW_BOND_PAC_SCHED_REG
FLOW_BOND_PAC_SCHED_MIN	Reserve for future use. Currently it is the same as FLOW_BOND_PAC_SCHED_REG
FLOW_BOND_CAPPED_RATE	Bond (capped, if any) coupon rate

See Also

- get_bond_flow_ex()

Enumerator

FLOW_BOND_PRINCIPAL_WRITEDOWN Principal writedown.

FLOW_BOND_RATE Bond (capped, if any) coupon rate (same as FLOW_BOND_CAPPED_RATE)

FLOW_BOND_FLT_INDEX Index of reference rate.

FLOW_BOND_INTEREST_DUE Interest calculated from coupon rate and outstanding balance.

FLOW_BOND_INTEREST_SHORTFALL Interest shortfall.

FLOW_BOND_UNCAPPED_RATE Uncapped coupon rate.

FLOW_BOND_CAPPED_COUPON Capped coupon rate.

FLOW_BOND_BASIS_SHORTFALL Basis shortfall.

FLOW_BOND_REIMB_LOSSES Reimbursed losses.

FLOW_BOND_DEFERRED_INTEREST Deferred interest.

FLOW_BOND_ATTACHMENT Attachment point projection, see MOODYS_SSFA_CALC for more info.

FLOW_BOND_DETACHMENT Detachment point projection, see MOODYS_SSFA_CALC for more info.

FLOW_BOND_SSFA_W SSFA W projection, see MOODYS_SSFA_CALC for more info.

FLOW_BOND_SSFA_KA SSFA Ka projection, see MOODYS_SSFA_CALC for more info.

FLOW_BOND_SSFA_KSSFA SSFA Kssfa projection, see MOODYS_SSFA_CALC for more info.

FLOW_BOND_SSFA_RW SSFA RW (risk weight) projection, see MOODYS_SSFA_CALC for more info.

FLOW_BOND_SSFA_KG SSFA Kg projection, see MOODYS_SSFA_CALC for more info.

FLOW_BOND_IMPLIED_LOSS Implied losses (SFW only)

FLOW_BOND_CURR_INS_INT_DUE Current interest due from insurance account. (SFW only)

FLOW_BOND_CURR_INS_INT_PAID Current interest paid from insurance account. (SFW only)

FLOW_BOND_CURR_INS_PRIN_DUE Current principal due from insurance account. (SFW only)

FLOW_BOND_CURR_INS_PRIN_PAID Current principal paid from insurance account.(SFW only)

FLOW_BOND_PAST_INS_INT_DUE Past interest due from insurance account. (SFW only)

FLOW_BOND_PAST_INS_INT_PAID Past interest paid from insurance account. (SFW only)

FLOW_BOND_PAST_INS_PRIN_DUE Past principal due from insurance account. (SFW only)

FLOW_BOND_PAST_INS_PRIN_PAID Past principal paid from insurance account. (SFW only)

FLOW_BOND_IO_BALANCE Balance flow of IO tranches.

FLOW_BOND_NONIO_BALANCE Balance flow of non-IO tranches.

FLOW_BOND_MONTH_END_ACCRUED_INT Interest accrued from last coupon payment date to the end of the month.

22.1.3.16 enum EXTENDED_FLOW_COLLATERAL_IDENTIFIER

Collateral cashflow identifiers

See Also

- get_collateral_flow_ex()

Enumerator

FLOW_COLLATERAL_BALANCE Balance (at the end) of current period.

FLOW_COLLATERAL_SCHED_PRINCIPAL Amount of the scheduled principal.

FLOW_COLLATERAL_PREPAYMENTS Amount of voluntary prepayment.

FLOW_COLLATERAL_DEFAULTS Amount of defaults brought forward to the end of liquidation period.

FLOW_COLLATERAL_LOSSES Amount of losses after property liquidation.

FLOW_COLLATERAL_LIQUIDATIONS Principal recovered after liquidation, same as FLOW_COLLATERA-L_PRIN_RECOVERIES.

FLOW_COLLATERAL_PRIN_RECOVERIES Principal recovered after liquidation, same as FLOW_COLLA-TERAL_LIQUIDATIONS.

FLOW_COLLATERAL_INTEREST Actual interest paid.

FLOW_COLLATERAL_REINVESTMENT Collateral reinvestment.

FLOW_COLLATERAL_CASH Total amount of cash available for distribution by payment rules, can be calculated as sum of scheduled principal, interest, prepayment, and recovery; for HELOCs the amount of draw amount is subtracted.

FLOW_COLLATERAL_BOND_VALUE Value of the collateral (set identical to FLOW_COLLATERAL_BALA-NCE)

FLOW_COLLATERAL_PO_BALANCE Sum of the balance portions of all discount loans in ratio strip POs.

FLOW_COLLATERAL_EXCESS_INTEREST Excessive interest (for ratio strip IOettes)

FLOW_COLLATERAL_DEL_30 Projections of 30 days of delinquency rates expressed in decimals (0.05 for 5%),CHS only.

FLOW_COLLATERAL_DEL_60 Projections of 60 days of delinquency rates expressed in decimals (0.05 for 5%),CHS only.

FLOW_COLLATERAL_DEL_90P Projections of 90+ days of delinquency rates expressed in decimals (0.05 for 5%),CHS only.

FLOW_COLLATERAL_PREPAY_PENALTY Amount paid as a result of application of payment penalty rules (points, yield maintenance, etc), mostly for CMBS.

FLOW_COLLATERAL_NEGATIVE_AMORT Amount of negative amortization during given period, ether 0 or absolute value of negative scheduled principal.

FLOW_COLLATERAL_BEG_BAL_INTEREST Interest calculated on the amount at the start of the period.

FLOW_COLLATERAL_PO_LOSSES PO losses (for ratio strip POs)

FLOW_COLLATERAL_DRAW_AMOUNT Amount of additional money drawn during the period (for HELOC)

FLOW_COLLATERAL_EXCESS_LOSS Excessive loss.

FLOW_COLLATERAL_STUDENT_LOAN_DELAYED_INTEREST Delayed interest for student loan, residual balance auto lease.

FLOW_COLLATERAL_AUTO_LEASE_RESIDUE_LOSS Auto lease residue losses.

FLOW_COLLATERAL_AUTO_LEASE_NET_RESIDUE_SCHED_FLOW Auto lease net residue (scheduled)

FLOW_COLLATERAL_CPR_1M CPR prepayment rate calculated from the voluntary prepayment amount.

FLOW_COLLATERAL_CDR_1M CDR default rate calculated from the defaulted principal amount.

FLOW_COLLATERAL_END_PERFORMING_BAL Performing balance at the end of the period.

FLOW_COLLATERAL_DEFAULTS_NO_DELAY Defaults without delay (same as FLOW_COLLATERAL_DEFAULTS for some cases)

FLOW_COLLATERAL_DELBAL_30 Ending balance of defaulted loans outstanding (not yet liquidated) for 30 - 59 days (SFW RMBS, CMBS, and Auto Deals)

FLOW_COLLATERAL_DELBAL_60 Ending balance of defaulted loans outstanding (not yet liquidated) for 60 - 89 days (SFW RMBS, CMBS, and Auto Deals)

FLOW_COLLATERAL_DELBAL_90P Ending balance of defaulted loans outstanding (not yet liquidated) for 90+ day (SFW RMBS, CMBS, and Auto Deals)

FLOW_COLLATERAL_SCHED_P_AND_I Scheduled principal and interest.

FLOW_COLLATERAL_PO_SCHED_PRIN Scheduled principal for PO strips.

FLOW_COLLATERAL_PO_PREPAYMENTS Prepayments for PO strips.

FLOW_COLLATERAL_PO_PRIN_RECOVERIES Principal recoveries for PO strips.

FLOW_COLLATERAL_PERFORMING_BALANCE Performing balance.

FLOW_COLLATERAL_IO_BALANCE Balance of IO strips.

FLOW_COLLATERAL_ACCRUED_INTEREST Interest accrued.

FLOW_COLLATERAL_ACCURED_INTEREST Typo, same as FLOW_COLLATERAL_ACCRUED_INTEREST, will be removed later.

FLOW_COLLATERAL_UNSCHED_PRINCIPAL Principal from default recovery and prepayment.

FLOW_COLLATERAL_PRINCIPAL Sum of unscheduled principal and scheduled principal.

FLOW_COLLATERAL_STUDENT_LOAN_BAL_SCH_AND_GRACE Balance of loans under school or grace period (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_BAL_DEFERMENT Balance of loans under deferment period (S-LABS)

FLOW_COLLATERAL_STUDENT_LOAN_BAL_FOREBEARANCE Balance of loans under forbearance period (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_BAL_REPAYMENT Balance of loans under repayment period (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_FFELP_INT_RECOVERY Interest reimbursement due to recovered default principal (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_ACCRUED_TO_BE_CAPITALIZED Cumulative accrued interest to be capitalized (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_CAPITALIZED_INT Capitalized interest amount which will be added into current balance (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_INT_LOSS Shortfall of due interest from defaulted principal (SL-ABS)

FLOW_COLLATERAL_STUDENT_LOAN_MONTHLY_ACCRUAL_SAP_PAYMENT Monthly SAP payment accrued (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_SAP_PAYMENT SAP payment with delay (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_MONTHLY_ACCRUAL_SAP_EXCESS_INT_PAID_TO_DOE Monthly excess interest accrued paid to department of education (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_SAP_EXCESS_INT_PAID_TO_DOE Excess interest paid to department of education (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_MONTHLY_ACCRUAL_SUBSIDY_PAYMENT Monthly subsidy payment accrued (SLABS)

FLOW_COLLATERAL_STUDENT_LOAN_SUBSIDY_PAYMENT Subsidy payment (SLABS)

FLOW_COLLATERAL_INTEREST_OF_BUYS Interest of Buys.

FLOW_COLLATERAL_REINVESTMENT_INCOME Reinvestment Income.

FLOW_COLLATERAL_LOSS_SEVERITY Loss Severity (CDOnet only)

FLOW_COLLATERAL_BALANCE_OF_BUYS Balance of Buys (CDOnet only)

FLOW_COLLATERAL_DEFAULT_OF_BUYS Defaulted of Buys (CDOnet only)

FLOW_COLLATERAL_LOSSES_OF_BUYS Losses of Buys (CDOnet only)

FLOW_COLLATERAL_RECOVERY_OF_BUYS Recovery of Buys (CDOnet only)

FLOW_COLLATERAL_SCHED_PRIN_OF_BUYS Scheduled principal of Buys (CDOnet only)

FLOW_COLLATERAL_TOTAL_PRIN_OF_BUYS Total principal of Buys (CDOnet only)

FLOW_COLLATERAL_UNSCHED_PRIN_OF_BUYS Unscheduled principal of Buys (CDOnet only)

FLOW_COLLATERAL_WAC Weighted average coupon.(SFW only)

22.1.3.17 enum MARKIT_POOL_HISTORY

Enumerator

MARKIT_POOL_HISTORY_CPR1M 1 Month CPR Rate

MARKIT_POOL_HISTORY_CPR3M 3 Month CPR Rate

MARKIT_POOL_HISTORY_CPR6M 6 Month CPR Rate

MARKIT_POOL_HISTORY_CPR12M 12 Month CPR Rate

MARKIT_POOL_HISTORY_CPR24M 24 Month CPR Rate

MARKIT_POOL_HISTORY_CPR_LIFE Life time CPR Rate.

MARKIT_POOL_HISTORY_PSA1M 1 Month PSA Rate

MARKIT_POOL_HISTORY_PSA3M 3 Month PSA Rate

MARKIT_POOL_HISTORY_PSA6M 6 Month PSA Rate

MARKIT_POOL_HISTORY_PSA12M 12 Month PSA Rate

MARKIT_POOL_HISTORY_PSA24M 24 Month PSA Rate

MARKIT_POOL_HISTORY_PSA_LIFE Life time PSA Rate.

MARKIT_POOL_HISTORY_DRAW1M 1 Month Draw Rate

MARKIT_POOL_HISTORY_DRAW3M 3 Month Draw Rate

MARKIT_POOL_HISTORY_DRAW6M 6 Month Draw Rate

MARKIT_POOL_HISTORY_DRAW12M 12 Month Draw Rate

MARKIT_POOL_HISTORY_DRAW_LIFE Life time Month Draw Rate.

22.1.3.18 enum MISSING_INTEREST_RATES_HANDLING

Enum for the action when some interest rates are missing.

See Also

set_missing_interest_rates_handling()

Enumerator

MISSING_INTEREST_RATES_USE_ZERO Use 0 for missing interest rate.

MISSING_INTEREST_RATES_TREAT_AS_ERROR Treat missing interest rate as error.

22.1.3.19 enum PA_MULTIPLIER_TYPE

Multiplier type for PA

New feature Subject to change

See Also

set_pa_multiplier()

Enumerator

PA_MULTIPLIER_PREPAY Prepay multiplier.

PA_MULTIPLIER_DEFAULT Default multiplier.

PA_MULTIPLIER_SEVERITY Severity multiplier.

NUM_PA_MULTIPLIER_TYPE Number of PA multiplier type.

22.1.3.20 enum PA_POOL_VECTOR_TYPE

Output vector type for PA.

New feature Subject to change

See Also

get_pa_vector()

Enumerator

PA_DQ_RATE_30 Percentage of outstanding balance that is 30-59 days past due, expressed in percent (20% expressed as 20).

PA_DQ_RATE_60 Percentage of outstanding balance that is 60 days past due, expressed in percent (20% expressed as 20).

PA_DQ_RATE_90 Percentage of outstanding balance that is 90 days or more past due, expressed in percent (20% expressed as 20).

PA_CPR Conditional Prepayment Rate, annualized, expressed in percent (20% expressed as 20).

PA_CDR Conditional Default Rate, annualized, expressed in percent (20% expressed as 20).

PA_SEVERITY Loss given default, expressed in percent (20% expressed as 20).

PA_PRINCIPAL_PAYMENT Percentage of the outstanding balance one month ago, paid in the last 30 days, expressed in percent (20% expressed as 20).

PA_SELLER_PERCENTAGE Percentage of the master trust owned by seller, expressed in percent (20% expressed as 20).

PA_YIELD Finance charge collections from borrowers divided by the pool principal balance at the end fo the prior period, annualized, expressed in percent (20% expressed as 20).

PA_CHARGEOFF Percentage of outstanding principal that is charged off in this month, expressed in percent (20% expressed as 20).

PA_PP_LOC_GROWTH Monthly growth rate, in percent, of the Unpaid Principal Balance of the Line of Credit portion of the pool, expressed in percent (20% expressed as 20).

PA_PP_TEN_GROWTH Monthly growth rate, in percent, of the Unpaid Principal Balance of the Tenure portion of the pool, expressed in percent (20% expressed as 20).

PA_PP_TER_GROWTH Monthly growth rate, in percent, of the Unpaid Principal Balance of the Term portion of the pool, expressed in percent (20% expressed as 20).

PA_PP_MTN_GROWTH Monthly growth rate, in percent, of the Unpaid Principal Balance of the Modified Tenure portion of the pool, expressed in percent (20% expressed as 20).

PA_PP_MTM_GROWTH Monthly growth rate, in percent, of the Unpaid Principal Balance of the Modified Term portion of the pool, expressed in percent (20% expressed as 20).

PA_REPAYMENT_RATE Percentage of outstanding balance that is due in this month, expressed in percent (20% expressed as 20).

PA_DEFERMENT_RATE Amount of receivables belonging to loans in which the obligor has been granted a deferment, divided by the current collateral balance, expressed in percent (20% expressed as 20).

PA_FORBEARANCE_RATE Amount of receivables belonging to loans in which the obligor has been granted a forbearance, divided by the current collateral balance, expressed in percent (20% expressed as 20).

PA_TOTAL_DQ_RATE Total delinquency rate, annualized, currently it is work the same as the 90 days of delinquency rates, expressed in percent (20% expressed as 20).

NUM_PA_POOL_VECTOR_TYPES Number of PA output vector type.

22.1.3.21 enum PREPAY_PENALTY_STRUCTURE

Enumerator

NOT_AVAILABLE not available

LOCKOUT_ONLY lockout only

POINTS_ONLY points only

YIELDMAINTENANCE_ONLY yieldmaintenance only

LOCKOUT_TO_POINTS lockout to points

YIELDMAINTENANCE_TO_POINTS yieldmatainenance to points

UNKNOWN_TO_POINTS unknown to points

22.1.3.22 enum PRICING_ANCHORS

Enum for the pricing anchors.

See Also

price_bond()

Enumerator

PRICE Price.

YIELD Yield.

DM Discount Margin.

22.1.3.23 enum STUDENT_LOAN_REPAY_TYPE

Enumerator

STUDENT_LOAN_REPAY_TYPE_FULL_DEFER	capitalize interest
STUDENT_LOAN_REPAY_TYPE_PRIN_DEFER	pay interest, balance flat

22.1.3.24 enum STUDENT_LOAN_STATE

Enumerator

STUDENT_LOAN_STATE_REPAY	Loan in the repayment status.
STUDENT_LOAN_STATE_DEFER	Loan in the deferment status.
STUDENT_LOAN_STATE_FORBEAR	Loan in the forbearance status.
STUDENT_LOAN_STATE_IN_SCHOOL	Loan in the school period.
STUDENT_LOAN_STATE_GRACE_PERIOD	Loan in the grace period.

22.1.3.25 enum TMP_DIR_TREATMENT

Can be used to indicate how temporary files are treated when the API exits.

See Also

set_tmp_dir_treatment()

Enumerator

TMP_DIR_DO_NOTHING	No extra cleanup will be performed.
TMP_DIR_REMOVE_THIS_PROCESS	Obsolete, no extra cleanup will be performed.
TMP_DIR_REMOVE_ALL	The API will try to remove all temporary files if possible.

22.1.4 Function Documentation

22.1.4.1 int CHASAPI calc_cashflow_offset_ex (void ∗ *tid,* const char ∗ *bondid,* const char ∗ *settlement_date,* int ∗ *months_offset,* int ∗ *days_accrued,* int ∗ *days_offset*)

Calculates the following with a given bond settlement date(settlement_date).:

- whole cashflow offsetting periods(months_offset),
- days of accrued interest(days_accured),
- offsetting days(days_offset) for long or short in the first period for a bond transaction.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
`in`	*bondid*	The ID/Name of a bond.
`in`	*settlement_date*	The transaction settlement date expressed as mm/dd/yy. May 3, 2001 would be expressed as 05/03/01
`out`	*months_offset*	Number of months to offset cashflows returned by run_deal_ex (see notes)
`out`	*days_accrued*	Days of accrued interest
`out`	*days_offset*	Days in the first period for a bond transaction is long (positive) or short (negative)

Return values

0	No Error
-1	Invalid Bond ID
-2	Invalid Settlement Date – often caused by invalid format

Note

- The first period in the cashflows is based on the date the deal is opened "as of". The buyer of a bond may not be entitled to all of the cashflows due to the settlement date of the transaction. The months_offset determines the first cashflow the buyer would be entitled to.
- The method only uses 30/360 day count convention for CHS deals, and uses different day count conventions(ACT/360, ACT/365, ACT/ACT, 30/360, 30/365,30E/360) for SFW/CDOnet deals.

Example:

```
*       // Determine information for evaluating transaction on bond QI
*       // settling on March 13th, 2005.
*       void * tid = NULL;
*       // Deal is already opened
*
*       int monthsOffset;
*       int daysAccrued;
*       int daysOffset;
*       int iR = calc_cashflow_offset_ex(tid, "QI", "03/13/05", &monthsOffset, &
       daysAccrued, &daysOffset);
*
```

22.1.4.2 long CHASAPI clean_up_call_ex (void ∗ *tid,* short *state,* long *loan_num,* BOOLYAN *set_sup_remic*)

Determines whether or not the deal will be run to call. The setting will apply to all underlying deals if set_sup_remic is TRUE. By default, deal will run into maturity instead of call.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*state*	Turns the call On (1) or Off (0).
in	*loan_num*	Reserved for future use. Always pass -1.
in	*set_sup_remic*	The clean-up call setting will apply to all underlying deals if set_up_remic is TRUE for CHS deals It will NOT apply to underlying deals if it is FALSE. This parameter is not implemented for SFW deals at the moment.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid dso identifier (tid)
-3	Error - Invalid loan number

Example:

```
*      set_input_path("C:/tmp/deals");
*      set_engine_preference(
      PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
*
*      void* tid=NULL;
*      CMO_STRUCT *pCmo= new CMO_STRUCT;
*      char deal_id[] = "AAM0401";
*      memset(pCmo, 0, sizeof(CMO_STRUCT));
*      memcpy(pCmo->dealid, deal_id, sizeof(deal_id) / sizeof(char));
*
*      open_deal_ex(tid, pCmo);
*      // Deal is already opened
*
*      // Set the cleanup call on and apply to any underlying deals
*      clean_up_call_ex(tid, 1, -1, true);
*
*      close_deal_ex(tid, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

- The state of clean_up_call_ex() must be set to on (1) in order for the call to be processed, even if the call criteria is met or overridden.
- For SFW deals, if called with state set to on (1), it is same as calling set_call_option() with type FORCED_EARLIEST_CALL.

See Also

- set_call_option()

22.1.4.3 long CHASAPI close_deal_ex (void ∗ *tid*, CMO_STRUCT ∗ *cmos*)

Closes the open deal and releases all the computational resources that was taken by the deal space. This function should be called between successive deals but does not need to be called between successive bonds in the same deal.

Since

0.9.0

Availability ALL

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*cmos*	This must be the CMO_STRUCT used in open_deal_ex().

Return values

>=0	No error
<0	Error

Example:

```
*   void* tid = NULL;
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   memset(pCmo, 0, sizeof(CMO_STRUCT));
*   char deal_id[] = "CHL06018";
*   int len = sizeof(deal_id) / sizeof(char);
*   memcpy(pCmo->dealid, deal_id, len);
*
*   set_input_path("the_input_path");
*   assert(0 == open_deal_ex(tid, pCmo));
*
*   close_deal_ex(tid, pCmo);
*   delete pCmo;
*   pCmo = NULL;
*
```

Note

Once the deal is closed, it must be re-opened and the scenario information must be re-set before it can be run again.

22.1.4.4 int CHASAPI create_deal_scenario_object (void ** *Tid,* short * *LogAction,* char * *LogFile,* BOOLYAN * *Debug*)

Creates a deal_scenario object (dso). Dso returns with Tid. Dso is a context identifier. This functions is used for creating an individual context. This is required to have multiple deals open at the same time or for multi-threading of deal scenarios. When users open multiple deals at the same time, they need to create separate context for each deal. Users need to create separate context for each thread if they are running multi-threading.

Since

0.9.0

Availability ALL

Each dso should have a distinct log file name and should be used in only one thread at a time. However, multiple dso's can be processed within the same thread.

Parameters

out	*Tid*	The deal/scenario object identifier. This will be passed to subsequent functions.
in	*LogAction*	Determines how messages are returned. See LogAction param of set_log_options() for more info.
in	*LogFile*	Fully qualified path and name of the log file.
in	*Debug*	Whether or not to produce detailed debug info. (irrelevant if LogAction = 0) See Debug param of set_log_options() for more info.

Return values

1	Created - log file name changed to be unique (underscore(s) appended at end)
0	No error
-1	Error - Invalid LogAction
-2	Error - Tid is invalid or other error
-3	Error - Log file could not be created
-4	Error - No dso available - maximum have been used

Example:

```
*   void* tid = NULL;
*   short logAction = 1;
*   char *logFile="log.txt";
*   create_deal_scenario_object(&tid, &logAction, logFile, true);
*
```

Note

All dso's must be released by calling release_deal_scenario_object() in order to release resources. The log/messages options can be changed with set_log_options()

See Also

set_log_options() release_deal_scenario_object()

22.1.4.5 char* CHASAPI dayt_to_str (DAYT *julian*, char * *temp*)

Converts DAYT date to string (mm/dd/yy), only for 30/360 day count convention.

Since

0.9.0

Availability ALL

Parameters

in	*julian*	The date in the DAYT format.
out	*temp*	A pre-allocated pointer of at least 9 characters for the date (mm/dd/yy).

Return values

date	in mm/dd/yy format

Example:

```
*      // Gets the interest start date of the bond the deal was
*      // opened with. pCmo is the pointer to the CMO_STRUCT.
*      char temp[9];
*      dayt_to_str(pCmo->bond.date,temp);
*
```

Note

This function works for 30/360 day count convention. For other conventions, use dayt_to_str_with_day_count() instead. At least 9 characters must be allocated for date.

Warning

The cutoff year for mm/dd/yy is 79, so yy 80 is 1980 and yy 79 is 2079.

See Also

- dayt_to_str_with_day_count - Allows the user to convert the DAYT format to a string date for different day count.

22.1.4.6 char∗ CHASAPI dayt_to_str_with_day_count (DAYT *julian*, char ∗ *temp*, const int *dayCount*)

Converts DAYT date to string (mm/dd/yy) with given day count convention.

Since

1.6.0

Availability ALL

Parameters

in	*julian*	The date in the DAYT format.
in	*dayCount*	The day count rule. The day count rule must be one of: • DAYCOUNT_ACTUAL_360 • DAYCOUNT_ACTUAL_365 • DAYCOUNT_ACTUAL_ACTUAL • DAYCOUNT_30_360 • DAYCOUNT_30_365 • DAYCOUNT_30_360E
out	*temp*	A pre-allocated pointer of at least 9 characters for the date (mm/dd/yy).

Return values

date	in mm/dd/yy format

Example:

```
*      // Gets the interest start date of the bond the deal was
*      // opened with. pCmo is the pointer to the CMO_STRUCT.
*      // tid is the pointer to the deal-scenario object (dso).
*      MARKIT_BOND_INFO bond_info;
*      int ret = get_bond_info_by_tranche(tid, NULL, pCmo->
     bond.stripped_id, &bond_info);
*      if(ret != 0)
*      {
*          //error handling
*      }
*
*      char temp[9];
*      dayt_to_str_with_day_count(pCmo->bond.date, temp, bond_info.
     day_count);
*
```

Note

At least 9 characters must be allocated for date.

Warning

The cutoff year for mm/dd/yy is 79, so yy 80 is 1980 and yy 79 is 2079.

See Also

- dayt_to_str - Allows the user to convert the DAYT format to a string date for,only use for the day cout is 30/360.

22.1.4.7 int CHASAPI deal_has_underlying_deal (void ∗ *tid*)

Indicates whether the specified deal has underlying deals(REMICS) in its assets.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

2	Underlying deal(s) and regular collateral
1	Underlying deal(s), no regular collateral
0	Regular collateral only
-1	Error - Deal not opened or the tid is invalid
-4	Error - No collateral
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "AL2010-A";
*       int len = sizeof(deal_id) / sizeof(cahr);
*       memcpy(pCmo->dealid, deal_id,len);
*
*       set_input_path("C:\\deals");
*       assert(0 == open_deal_ex(tid, pCmo));
*       // Deal is opened.
*
*       assert(0 == deal_has_underlying_deal(tid));
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.8 void CHASAPI enable_same_deal_multithreading (int *flag*)

Enables the multithreading protection for the same deal.

Since

1.2.0

Availability ALL

Parameters

in	*flag*	If 1, will enable the same deal protection.

Returns

None

Example:

```
*     enable_same_deal_multithreading(1);
*
```

22.1.4.9 int CHASAPI get_agency_pool_prefix (void ∗ *tid*, char ∗ *pool_prefix*)

Retrieves the pool prefix for agency.

Since

2.7.0

Availability CHS

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*pool_prefix*	A pre-allocated pointer of at least 10 characters for the pool_prefix value get from agency LPD file.

Return values

0	No error
-1	Error: Deal not open
-2	Error: could not find pool prefix
-99	Error: Invalid dso identifier (tid) or other errors, please see details by calling get_-deal_error_msg()

Example:

```
*      void* pDeal = NULL;
*      //Deal has been opened
*
*      char pool_prefix[10];
*      int iRet = get_agency_pool_prefix(pDeal, pool_prefix);
*
```

22.1.4.10 MARKIT_POOL_INFO∗ CHASAPI get_average_collat (void ∗ *tid*, void ∗ *collat_iterator*, int *group_number*)

Returns the average collateral for this deal.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*collat_iterator*	Pointer to collateral iterator returned by calling obtain_collat_iterator().
in	*group_number*	Collateral group number, 0 for deal level.

Returns

Averaged collateral information.

Example:

```
*       void* tid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"AL2010-A");
*
*       int ret = open_deal_ex(tid,&deal);
*       if(ret == 0)
*       {
*           void* iter = obtain_collat_iterator(tid, 0);
*           MARKIT_POOL_INFO* aver_pool = get_average_collat(tid,iter,0);
*           assert(aver_pool->periodicity == deal.periodicity);
*
*           close_deal_ex(tid,&deal);
*       }
*
```

See Also

- get_average_collat_for_managed_code()
- get_average_collat_by_bond()

22.1.4.11 MARKIT_POOL_INFO∗ CHASAPI get_average_collat_by_bond (void ∗ *tid,* void ∗ *collat_iterator,* const char ∗ *bondid*)

Returns a pointer to MARKIT_POOL_INFO which holds the average value of the related poolgroup(s) of the bond specified.

Since

1.5.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*collat_iterator*	Pointer to collateral iterator returned by calling obtain_collat_iterator().
in	*bondid*	The name of the bond whose related collateral poolgroups are to be averaged.

Returns

Pointer to MARKIT_POOL_INFO that holds the average value of the poolgroup(s) related to the bond specified

Note

To get the related poolgroup(s) of a bond, call view_bond_coll_groups(). See documentation for MARKIT_P-OOL_INFO for aggregation/averaging logic.

Example:

```
*   void* tid=NULL;
*   CMO_STRUCT cmo;
*   memset(&cmo, 0, sizeof(CMO_STRUCT));
*   strcpy(cmo.dealid,"ACE06NC1");
*
*   open_deal_ex(tid,&cmo);
*
*   std::vector<CCMO_BONDS_S> pBonds(cmo.num_bonds);
*   view_all_bonds_ex(tid, &pBonds.front());
*
*   void* coll_it = obtain_collat_iterator(tid, 0);
*   if (NULL != coll_it)
*   {
*       // get the average of related pools of the first bond
*       MARKIT_POOL_INFO* related_pool_info =
      get_average_collat_by_bond(tid, coll_it, pBonds[0].stripped_id);
*   }
*
*   close_deal_ex(tid,&deal);
*
```

See Also

- get_average_collat()
- view_bond_coll_groups()

22.1.4.12 int CHASAPI get_average_collat_for_managed_code (void ∗ *tid,* void ∗ *collat_iterator,* int *group_number,* MARKIT_POOL_INFO ∗ *usr_pool,* CCMO_ARM_INFO ∗ *arm,* MARKIT_PAYMENT_SCHEDULE ∗ *sched,* MARKIT_PREPAY_PENALTY *prepayPenalty[ ],* int *sizeOfPpenArray,* int ∗ *hasArm,* int ∗ *hasSched,* int ∗ *hasPpen*)

This function is for managed code and gets the average collateral information using iterator obtained from calling obtain_collat_iterator(). All structures should have been allocated by the caller. There is no memory allocated by SDK.

Since

1.4.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*collat_iterator*	Pointer to collateral iterator returned by calling obtain_collat_iterator().
in	*group_number*	Collateral group number, 0 for deal level.
in	*sizeOfPpenArray*	Periods of prepayment penalty data requested.
out	*usr_pool*	The user allocated buffer to hold pool data.
out	*arm*	The user allocated buffer to hold adjustable rate information of usr_pool.
out	*sched*	The user allocated buffer to hold payment schedule data of usr_pool.
out	*prepayPenalty*	A pointer to a client-allocated array of MARKIT_PREPAY_PENALT in which the prepayment penalty data of usr_pool will be stored.

out	*hasArm*	Set to 1 if arm is set.
out	*hasSched*	Set to 1 if sched is set.
out	*hasPpen*	Just set to 0.

Return values

0	Collateral is successfully loaded or collat_iterator is NULL.
-1	usr_pool should be allocated by caller
-2	arm should be allocated by caller
-3	sched should be allocated by caller
-4	hasArm should be allocated by caller
-5	hasSched should be allocated by caller
-6	hasPpen should be allocated by caller
-99	Error, Invalid dso identifier (tid) or other errors, please see details by calling get_-deal_error_msg()

Example:

```
     *      void * tid = NULL;
     *      //Deal has opened
     *
     *      void*coll_it=obtain_collat_iterator(tid,0);
     *      if(coll_it==0)
     *      {
     *       std::cout<<"Failure to start collat iteration"<<get_deal_error_msg(tid)<<
            std::endl;
     *          return;
     *      }
     *
     *      MARKIT_POOL_INFO pool;
     *      CCMO_ARM_INFO arm;
     *      MARKIT_PAYMENT_SCHEDULE shed;
     *      MARKIT_PREPAY_PENALTY prepayPenalty[10];
     *      int has_Arm;
     *      int has_Sched;
     *      int hasPpen;
     *      //Get pool group 1 average info
     *      int iret = get_average_collat_for_managed_code(tid,coll_it,1,&
            pool,&arm,&sched,&prepayPenalty,10,&has_Arm,&has_Sched,&hasPpen);
     *      if (iret < 0)
     *      {
     *          //Error handling
     *      }
     *      //do what you want with average pool
     *
     *      memset(&pool, 0, sizeof(MARKIT_POOL_INFO));
     *      memset(&arm, 0, sizeof(CCMO_ARM_INFO));
     *      memset(&sched, 0, sizeof(MARKIT_PAYMENT_SCHEDULE));
     *      memset(prepayPenalty, 0, 10*sizeof(MARKIT_PREPAY_PENALTY));
     *      has_Arm = 0;
     *      has_Sched = 0;
     *      hasPpen = 0;
     *      //Get deal level average collateral info
     *      iret = get_average_collat_for_managed_code(tid,coll_it,0,&pool,&
            arm,&sched,&prepayPenalty,10,&has_Arm,&has_Sched,&hasPpen);
     *      if (iret < 0)
     *      {
     *          //Error handling
     *      }
     *      //do what you want with average pool
     *
```

See Also

get_average_collat()

22.1.4.13 int CHASAPI get_bond_band (void ∗ *tid,* const char ∗ *bondid,* double ∗ *pricing_wal,* double ∗ *low,* double ∗ *high*)

Retrieves the band information for the bond (pricing WAL, low speed and high speed). If NULL is passed for any item, that item will not be returned.

Since

1.1.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
out	*pricing_wal*	The weighted average life of the bond when the deal is priced.
out	*low*	The low collar speed for the band.
out	*high*	The high collar speed for the band.

Return values

0	No error
<0	Error: Check that deal is open and the bondid is valid

Example:

```
*       void * tid = NULL;
*       // Deal is already open
*
*       // Get pricing WAL only for bond A1
*       double wal;
*       int iR = get_bond_band(tid, "A1", &wal, NULL, NULL);
*
```

22.1.4.14 void CHASAPI get_bond_by_index_ex (void * *tid,* CCMO_BONDS_S * *b,* long *index*)

Finds the bond for the specified index (0-based).

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*index*	0-based index number of bond
in,out	*b*	Pointer to a CCMO_BONDS_S structure allocated by the calling program.

Returns

None. The bond stripped_id is set to ERROR if the bond index was invalid or the deal not open.

Example:

```
*       void * tid = NULL;
*       // Deal is already open
*
*       // Get a pointer to the first bond
*       CCMO_BONDS_S bond={};
*       get_bond_by_index_ex(tid, &bond, 0);
*
```

22.1.4.15 void CHASAPI get_bond_by_name_ex (void ∗ *tid,* CCMO_BONDS_S ∗ *b,* const char ∗ *id*)

Finds the bond for the specified bond name.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*id*	Name of the bond
out	*b*	Pointer to a CCMO_BONDS_S structure allocated by the calling program.

Returns

None. The bond stripped_id is set to ERROR if the bond was not found or the deal not open.

Note

The stripped_id in the CCMO_BONDS_S structure is set to ERROR if the requested bond does not exist in the deal.

Example:

```
*       void * tid = NULL;
*       // Deal is already open
*
*       // Get a pointer to the bond named A1
*       CCMO_BONDS_S *bond;
*       int iR = get_bond_by_name_ex(tid, bond, "A1");
*
```

22.1.4.16 char∗ get_bond_cf_date (int *per,* char ∗ *date,* void ∗ *tid,* const char ∗ *bondid*)

Returns the payment date corresponding to the requested period and bond. This method returns values from M-ARKIT_BOND_CASHFLOW.dates, which are adjusted based on the bond-level BUS_RULES set up in the deal file.

Since

1.5.0

Availability CHS, SFW

Precondition

run_deal_ex() has been called, and get_longest_ex has been called to get the max size of valid cf dates.

Parameters

per	The period whose cashflow date is being requested. Must be between 0 and MAX_PERIODS - 1.

date	A pointer to a null-terminated string (YYYYMMDD). This must be pre-allocated with at least 11 characters.
tid	The deal/scenario object identifier. Null if using non-thread-safe calls.
bondid	The name of the bond whose date is being requested.

Return values

Pointer	A pointer to a string. The date in YYYYMMDD format.
-2	Error - Period(per) is out of range.
Other	Error - Call get_deal_error_msg() for details.

Note

- This is a variation of the existing deal-level get_cf_date() method. Different bonds can have different BUS_RULES. This method returns the adjusted bond-level payment dates, whereas get_cf_date() returns the unadjusted deal-level payment dates.
- For more information about BUS_RULE, please refer to MARKIT_BOND_INFO.

Example:

```
*   void* tid = NULL;
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   memset(pCmo, 0, sizeof(CMO_STRUCT));
*
*   set_input_path("C:\\");
*   // open deal
*   if(0 != open_deal_ex(tid, pCmo))
*   {
*       delete pCmo;
*       pCmo = NULL;
*       return;
*   }
*   // run deal
*   if(0 != run_deal_ex(tid, pCmo))
*   {
*       delete pCmo;
*       pCmo = NULL;
*       return;
*   }
*
*   // get cf_date of period 9 in bond A1
*   char cf_date[11] = {0};
*   get_bond_cf_date(9, cf_date, tid, "A1");
*
*   // close deal
*   close_deal_ex(tid, pCmo);
*   delete pCmo;
*   pCmo = NULL;
*
```

22.1.4.17 short CHASAPI get_bond_day_cal_cur_ex (void ∗ *tid,* const char ∗ *bondid,* BOOLYAN *use_code,* char ∗ *day_count,* char ∗ *bus_rules,* char ∗ *currency*)

Retrieves the day count convention, business day/calendar rules, and currency for the specified bond. If use_code is true, the codes will be returned. If use_code is false, the description will be returned. This function requires that the file SDKCODES.TXT be in your deal directory. set_input_path() must have been called before this function if the description is requested (use_code is false).

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
in	*use_code*	If true, the actual code will be returned. If false, descriptive information is returned.
out	*day_count*	The day count rule used to compute interest (such as 30/360). See "Day Counts" paragraph for detail. This should be allocated by the user with at least 26 characters.
out	*bus_rules*	The rules for determining the payment date (such as next business day if the payment date is not a business day). See "Calendar rules" paragraph for detail. This should be allocated by the user with at least 26 characters.
out	*currency*	The currency of the bond. See "Currencies" paragraph for detail. This should be allocated by the user with at least 26 characters.

Return values

0	No error
-1	Deal not open
-2	Bond not found
-99	Invalid dso identifier (tid) or other errors, please see details by calling get_deal_-error_msg()

Note

If the code description file (SDKCODES.TXT) is not in your deal directory, or set_input_path() has not been called, all description requests will return "Invalid" as the description.

Day Counts

Day counts affect how interest is accrued for a transaction. The following day count codes are used.

Code	Description	Notes
1	Act/360	Actual days in period, 360 days in year
2	Act/365	Actual days in period, 365 days in year
3	Act/Act	Actual days in period, actual days in year
4	30/360	30 days in month, 360 days in year
5	30E/360	30 days in month, 360 days in year, last day in Feb = 30th.

Calendar rules

Calendar rules affect when a transaction settles. The following calendar rule codes are used.

Code	Description	Comments
A	Next Bus Day	If settlement is not a business day, settle on the next business day.
B	Next Bus Day In Month	If settlement is not a business day, settle on the next business day if it is in the current month. If not, settle on the prior business day.

C	Next BD Aft Serv Remit	Next business day after the master servicer remittance date.
P	Prev Bus Day	If settlement is not a business day, settle on the previous business day.
N	No Adjustment	Do not adjust for business days.

Currencies

Code	Description
E	Euros
F	Swiss Francs
K	Swedish Kronors
L	British Pounds
U	US Dollars
R	Russian Rubles

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       // Get the codes for the day counts, business rules and currency
*       // bond A1
*       char dayCount[26];
*       char busRules[26];
*       char currency[26];
*       assert(0 == get_bond_day_cal_cur_ex(pDeal, "A1", 1, dayCount, busRules,
      currency));
*
*       // Get the descriptions for the day counts, business rules and currency
*       // for bond A1
*       assert(0 == get_bond_day_cal_cur_ex(pDeal, "A1", 0, dayCount, busRules,
      currency));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.18 double* CHASAPI get_bond_flow_ex (void * *tid*, const char * *bondid*, int *flow_identifier*)

Returns a pointer to a vector of doubles containing the specified bond cashflow, the vector size is [MAX_PERIODS].

The balance, principal and interest for the specified bond can also be returned in CMO_STRUCT (double principal[MAX_PERIODS], double balance[MAX_PERIODS] and double interest[MAX_PERIODS]). For detail information, please refer to CMO_STRUCT.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The bond for which flows are requested.
in	*flow_identifier*	Identifies the requested cash flow. Must be one of the bond cashflow identifiers (see EXTENDED_FLOW_BOND_IDENTIFIER).

Return values

NULL	Error - Call get_deal_error_msg().
0	get bond flow successfuly.
OTHER	Pointer to the vector of cashflows.

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT cmo;
*       memset(&cmo,0,sizeof(CMO_STRUCT));
*       strcpy(cmo.dealid,"ACE06NC1");
*
*       open_deal_ex(ptid,&cmo);
*       run_deal_ex(ptid,&cmo);
*       // Deal is already opened successfully.
*       // Deal is already run successfully.
*
*       // Get bond flow of balance.
*       double* pbond_balance = get_bond_flow_ex(ptid,cmo.bond.
       stripped_id,FLOW_BOND_BALANCE);
*
*       close_deal_ex(ptid,&cmo);
*
```

Note

- Call set_deal_calc_level() with parameter CALC_LEVEL_FULL to retrieve FLOW_BOND_INTEREST_-SHORTFALL, FLOW_BOND_BASIS_SHORTFALL, FLOW_BOND_CAPPED_COUPON and FLOW_-BOND_REIMB_LOSSES via get_bond_flow_ex(),after open_deal_ex() but before run_deal_ex()
- If error message is 'Need to call open_deal_ex first!!!', it means deal is not opened.
- If error message is 'bondid provided is NULL.', it means input parameter bondid is NULL.
- If error message is 'Unable to find tranche ErrBondid', it means input parameter, bondid is invalid.

See Also

- get_bond_flow_ex can be used to get specified bond cashflows. It returns projected bond cashflows specified by flow_identifier.
- get_bond_flow_ex1 can return specified bond cashflows for all bond flow identifiers.
- get_bond_flow_ex1_for_managed_code is similar to get_bond_flow_ex1, but it needs the user to allocate data structure before the call.

22.1.4.19 MARKIT_BOND_CASHFLOW∗ CHASAPI get_bond_flow_ex1 (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *bondid*)

Returns a pointer to MARKIT_BOND_CASHFLOW structure with bond cashflow.

This function is different from get_bond_flow_ex() by the fact that it might have one historical payment to which many zero delay floater bonds are entitled to, while get_bond_flow_ex() returns only projected cashflows.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier.
in	*reremic_deal_id-_or_null*	0 for parent deal or name of the child deal.
in	*bondid*	The tranche name.

Return values

Null	Error - call get_deal_error_msg()
Not-Null	Pointer to MARKIT_BOND_CASHFLOW structure with all the cashflow information for the deal. Please refer to MARKIT_BOND_CASHFLOW for details.

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT cmo;
*       memset(&cmo,0,sizeof(CMO_STRUCT));
*       strcpy(cmo.dealid,"ACE06NC1");
*
*       open_deal_ex(ptid,&cmo);
*       run_deal_ex(ptid,&cmo);
*       // Deal is already opened successfully.
*       // Deal is already run successfully.
*
*       // Get bond flow of balance
*       MARKIT_BOND_CASHFLOW* pbond_cash_flow =
      get_bond_flow_ex1(ptid,cmo.bond.stripped_id);
*       if(NULL == pbond_cash_flow)
*       {
*           const char*err_msg=get_deal_error_msg(ptid);
*           if(NULL!=err_msg)
*           {
*               std::cout<<"Fail to call get_bond_flow_ex1:"<<err_msg<<std:endl;
*           }
*       }
*
*       close_deal_ex(ptid,&cmo);
*
```

Note

- Call set_deal_calc_level() with CALC_LEVEL_FULL setting to get the value for the following fields: double∗interest_due, double∗interest_shortfall, int∗accrual_days, double∗uncapped_rate) in MARKIT-_BOND_CASHFLOW after calling open_deal_ex() but before run_deal_ex()

See Also

- get_bond_flow_ex1_for_managed_code can return specified bond cashflows for both top level deals and underlying deals. It needs user allocate MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE structure before the call.
- get_bond_flow_ex1 returns a pointer to the bond cashflows of MARKIT_BOND_CASHFLOW structure.
- get_bond_flow_ex can be used to get specified bond cashflows for both top level deals and underlying deals. It returns projected bond cashflows specified by flow_identifier.

22.1.4.20 int CHASAPI get_bond_flow_ex1_for_managed_code (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *bondid,* MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE ∗ *cf*)

Return specified bond cashflow of MARKIT_BOND_CASHFLOW structure via output parameter "cf". It is a variation of function get_bond_flow_ex1(), which returns a pointer. The difference is that the structure MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE needs to be allocated by the user when this function is called. All data will be stored in static memory.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	If reremic deal, this is the id, otherwise null
in	*bondid*	The bond id
out	*cf*	Pointer to the structure holding bond cash flow data

Return values

0	Success
-99	Error, invalid dso identifier (tid) or other errors, please see details by calling get_-deal_error_msg()
-1	Error, deal is not opened
<0	Others Error, for details call get_deal_error_msg()

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT cmo;
*       memset(&cmo,0,sizeof(CMO_STRUCT));
*       strcpy(cmo.dealid,"ACE06NC1");
*
*       open_deal_ex(ptid,&cmo);
*       run_deal_ex(ptid,&cmo);
*       // Deal is already opened successfully
*       // Deal is already run successfully
*
*       // Get bond flow of balance
*       MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE MarkitBondCf;
*       memset(&MarkitBondCf,0,sizeof(MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE
      ));
*       int iret = get_bond_flow_ex1_for_managed_code(ptid, NULL, cmo.
      bond.stripped_id, &MarkitBondCf);
*       if(0!=ret)
*       {
*           const char*err_msg=get_deal_error_msg(ptid);
*           if(NULL!=err_msg)
*           {
*               std::cout<<"Fail to call get_bond_flow_ex1_for_managed_code:"<<err_msg<<std::endl;
*           }
*       }
*
*       close_deal_ex(ptid,&cmo);
*
```

Note

- Call set_deal_calc_level() with CALC_LEVEL_FULL, to get value for the following fields : double interest-_due[MAX_PERIOD], double interest_shortfall[MAX_PERIODS], double uncapped_rate[MAX_PERIO-DS] in MARKIT_BOND_CASHFLOW.

See Also

- get_bond_flow_ex1_for_managed_code can return specified bond cashflows for both top level deals and underlying deals. It needs user allocate MARKIT_BOND_CASHFLOW_FOR_MANAGED_CODE structure before the call.

- get_bond_flow_ex1 returns a pointer to the bond cashflows of MARKIT_BOND_CASHFLOW structure.
- get_bond_flow_ex can be used to get specified bond cashflows for both top level deals and underlying deals. It returns projected bond cashflows specified by flow_identifier.

22.1.4.21 long CHASAPI get_bond_index_ex (void ∗ *tid,* const char ∗ *id*)

Finds the 0-based index for the requested bond.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*id*	A pointer to the name of the bond.

Return values

>=0	The zero-based (0) index of the bond
-1	Deal not open
-2	Bond not found
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void * tid = NULL;
*       // Deal is already open
*
*       int index = get_bond_index_ex(tid, "A2B");
*
```

22.1.4.22 int CHASAPI get_bond_info_by_index (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int *index,* MARKIT_BOND_INFO ∗ *bond_info*)

Gets information for any single bond in the deal based on its index or position in the capital structure. The index values start from 1.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	If reremic deal, this is the ID, otherwise null.
in	*index*	The 1-based index of the bond in the array of bonds.
out	*bond_info*	A pointer to the structure holding the bond information.

Return values

0	Success
-1	Deal not open
-3	Invalid bond index
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened.
*
*       MARKIT_DEAL_INFO dealInfo;
*       memset(&dealInfo, 0, sizeof(MARKIT_DEAL_INFO));
*       get_deal_info(ptid,0,&dealInfo);
*
*       for(int i =1;i<= dealInfo.num_bonds; i++)
*       {
*           MARKIT_BOND_INFO bi;
*           get_bond_info_by_index(ptid, NULL, i, &bi);
*       }
*
*       close_deal_ex(ptid,&deal);
*
```

See Also

- get_bond_info_by_tranche()
- get_bond_index_ex()

Warning

get_bond_info_by_index() uses a 1-based index whereas get_bond_index_ex() returns a 0-based index.

22.1.4.23 int CHASAPI get_bond_info_by_tranche (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *bondid,* MARKIT_BOND_INFO ∗ *bond_info*)

Get the bond info from its tranche name.

Since

1.1.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	If reremic deal, this is the id, otherwise null
in	*bondid*	The bond tranche name
out	*bond_info*	Pointer to the structure holding bond info

Return values

0	Success
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void * ptid = NULL;
*       // Deal is already open.
*
*       std::vector<CCMO_BONDS_S> pBonds(pCmo->num_bonds);
*       view_all_bonds_ex(ptid, &pBonds.front());
*       for(int i = 0; i <= pBonds.size(); i++)
*       {
*           MARKIT_BOND_INFO bi;
*           get_bond_info_by_tranche(ptid, NULL, pBonds[i].stripped_id, &bi)
*       }
*
```

See Also

- get_bond_info_by_index()

22.1.4.24 short CHASAPI get_bond_misc_ex (void ∗ *tid*, const char ∗ *Bond*, BOOLYAN ∗ *IsSeg*, BOOLYAN ∗ *IsMACR*, BOOLYAN ∗ *IsPO*)

Get additional information on a bond (segment, MACR or PO).

Since

1.2.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*Bond*	A pointer to the name of the bond.
out	*IsSeg*	If non-zero, the bond is a segment bond. Otherwise it is not.
out	*IsMACR*	If non-zero, the bond is a MACR bond. Otherwise it is not.
out	*IsPO*	If non-zero, the bond is a PO bond. Otherwise it is not.

Return values

0	No error
-1	Deal not open
-2	Bond not found
-99	Invalid dso identifier (tid) or other errors, please see details by calling get_deal_-error_msg()

Example:

```
*       void * ptid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       // Pre-condition: Deal is already opened
*
*       BOOLYAN isSeg = 0;
*       BOOLYAN isMacr = 0;
*       BOOLYAN isPo = 0;
*       int ret = get_bond_misc_ex(ptid, pCmo->bond.
      stripped_id, &isSeg, &isMacr, &isPo);
*       if(ret < 0)
*       {
*             //error handle
*       }
*
```

See Also

- get_reremic_bond_misc()

22.1.4.25 int CHASAPI get_bond_payment_group (void ∗ *tid,* const char ∗ *bondid,* char ∗ *group_names[]*)

This method gets the payment group(s) name for the specified bond.

Since

2.5.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
out	*group_names*	A pointer to a client-allocated array of character strings in which the names of the groups will be stored. 10 characters should be allocated for each string. Only filled in if it is not NULL. If this parameter is NULL, the number of payment groups will return.

Return values

>=0	Success. Number of payment groups.
-1	Deal not open.
-2	Bond not found.
-99	Error, call get_deal_error_msg() for details.

Example:

```
*       void* pDeal = NULL;
*       // deal has been opened
*
*       int group_num = get_bond_payment_group(pDeal, "A1", NULL);
*       if(group_num > 0)
*       {
*           std::vector<char> name_buf(group_num*10);
*           std::vector<char*> names(group_num);
*           for(int i = 0; i<group_num; i++ )
*               names[i]=&name_buf[i*10];
*           group_num = get_bond_payment_group(pDeal, "A1", &names.front());
*       }
*
```

22.1.4.26 int CHASAPI get_bond_rate_determination_date (void ∗ *tid,* const char ∗ *bondid,* int ∗ *determination_date*)

Retrieves the date which two business days prior to the current accrual begin date of the bond.

Since

2.7.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
out	*determination_-date*	The weighted average life of the bond when the deal is priced.

Return values

0	No error
<0	Error: Check that deal is open and the bondid is valid

Example:

```
*      void* pDeal = NULL;
*      //Deal has been opened
*
*      int rateDeterminDate = 0;
*      int iRet = get_bond_rate_determination_date(pDeal, "NL", &
     rateDeterminDate);
*
```

22.1.4.27 char∗ get_cf_date (int *per,* char ∗ *date,* void ∗ *tid*)

Returns the deal level payment date corresponding to the requested period.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called, and get_longest_ex has been called to get the max size of valid cf dates.

Parameters

in	*per*	The period whose cashflow date is being requested. Must be between 0 and MAX_PERIODS - 1.

out	*date*	A pointer to a null-terminated string (YYYYMMDD). This must be pre-allocated with at least 11 characters.
in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

'YYYYMMDD'	The payment date in YYYYMMDD format. The first period stores the last payment date of the current deal update.
'-2'	Period(per) is out of range. Call get_deal_error_msg() for details.
'Error'	Other errors. Call get_deal_error_msg() for details.

Note

If get_cf_date() returns error code string instead of valid date string, the user should call get_deal_error_msg() to get detail error message.

Example:

```
*   set_input_path("C:/Deals");
*   set_engine_preference(PICK_CHS_ENGINE_FOR_MAPPED_DEALS
      );
*
*   void* tid=NULL;
*   CMO_STRUCT cmo={};
*   strcpy(cmo.dealid,"AAM0401");
*   open_deal_ex(tid, &cmo);
*   // Deal is opened
*
*   int longest=get_longest_ex(tid);
*   char cf_date[11]={};
*   for (int i=0; i<=longest; ++i)
*   {
*       char* msg=get_cf_date(i, cf_date, tid);
*       std::cout<< msg << std::endl;
*   }
*
*   close_deal_ex(tid, &cmo);
*
```

22.1.4.28 int CHASAPI get_cleanup_call_ex (void * *tid,* char * *CallDate,* double * *CallPct,* int * *CallPctCalc*)

Retrieves information about the deal call.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The tid identifier.
out	*CallDate*	The earliest call date (mm/dd/yy or mm/dd/yyyy). The calling program must allocate this with at least eleven (11) characters.
out	*CallPct*	The required percent for the deal to be callable.
out	*CallPctCalc*	Type of call balance. Should be the pointer to one of CLEAN_UP_CALL_BALANCE_TYPE.

Return values

get_cleanup_call_ex	Type of cleanup call
CLEANUP_CALL_NONE	No cleanup call
CLEANUP_CALL_DATE	Callable by date
CLEANUP_CALL_PERCE-NT	Callable by percent
CLEANUP_CALL_EITHER	Callable by date or percent
CLEANUP_CALL_BOTH	Callable by date and percent
-1	Deal not open
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       char callDate[11] = "03/15/16";
*       double callPct = 0.1;
*       int callPctCalc;
*       int rVal =get_cleanup_call_ex(tid, callDate, &callPct, &callPctCalc);
*
```

22.1.4.29 double∗ CHASAPI get_collateral_flow_ex (void ∗ *tid,* long *group_number,* int *flow_identifier*)

Returns a pointer to a vector of doubles containing the specified collateral cashflow, the vector size is MAX_PERI-ODS.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*group_number*	The collateral group for which cashflows are requested, 0 for total (deal level).
in	*flow_identifier*	Identifies the requested cash flow. Must be one of the collateral cashflow identifiers (see EXTENDED_FLOW_COLLATERAL_IDENTIFIER).

Return values

NULL	Error - Call get_deal_error_msg() for detail.
OTHER	Pointer to the vector of cashflows

Note

Call set_deal_calc_level() with parameter CALC_LEVEL_FULL after open_deal_ex() but before run_deal_ex() to retrieve the following collateral flow identifiers:

- FLOW_COLLATERAL_PO_BALANCE,
- FLOW_COLLATERAL_PO_LOSSES,
- FLOW_COLLATERAL_EXCESS_INTEREST,
- FLOW_COLLATERAL_NEGATIVE_AMORT,
- FLOW_COLLATERAL_DRAW_AMOUNT,
- FLOW_COLLATERAL_BEG_BAL_INTEREST,

- FLOW_COLLATERAL_SCHED_P_AND_I,
- FLOW_COLLATERAL_PO_SCHED_PRIN,
- FLOW_COLLATERAL_PO_PREPAYMENTS,
- FLOW_COLLATERAL_PO_PRIN_RECOVERIES,
- FLOW_COLLATERAL_PERFORMING_BALANCE,
- FLOW_COLLATERAL_IO_BALANCE,
- FLOW_COLLATERAL_CPR_1M.
- FLOW_COLLATERAL_CDR_1M.
- FLOW_COLLATERAL_DELBAL_30,
- FLOW_COLLATERAL_DELBAL_60,
- FLOW_COLLATERAL_DELBAL_90P.

Example:

```
*    void*ptid=NULL;
*    CMO_STRUCT cmo;
*    memset(&cmo, 0, sizeof(CMO_STRUCT));
*    strcpy(cmo.dealid,"ACE06NC1");
*
*    open_deal_ex(ptid,&cmo);
*    run_deal_ex(ptid,&cmo);
*    // Deal is already opened successfully.
*    // Deal is already run successfully.
*
*    // Get collateral flow of balance.
*    double*pcollat_flow=get_collateral_flow_ex(ptid, 1,
       FLOW_COLLATERAL_BALANCE);
*    if(NULL==pcollat_flow)
*    {
*        const char*err_msg=get_deal_error_msg(ptid);
*        if(NULL !=err_msg)
*        {
*            std::cout<<"Fail to call get_collateral_flow_ex:"<<err_msg<<std::endl;
*        }
*    }
*
*    close_deal_ex(tid, &cmo);
*
```

See Also

- get_collateral_flow_ex() is used to get specified collateral cashflow. It returns projected collateral cashflows specified by flow_identifier.
- get_collateral_flow_ex1() returns the specified group collateral cashflows. It needs user to allocate the MARKIT_COLLAT_CASHFLOW data structure.

22.1.4.30 int CHASAPI get_collateral_flow_ex1 (void ∗ *tid,* int *group_number,* const char ∗ *reremic_deal_id_or_null,* MARKIT_COLLAT_CASHFLOW ∗ *cf*)

Populates the user allocated structure of type MARKIT_COLLAT_CASHFLOW with collateral cash flow data and deal settings. The dynamic memory within the structure MARKIT_COLLAT_CASHFLOW will be allocated by the SDK.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Return values

0	SUCCESS
Others	Error - check the message for details

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*group_number*	The collateral group for which cashflows are requested, 0 for total (deal level).
in	*reremic_deal_id-_or_null*	Reremic deal ID or null if not reremic deal.
out	*cf*	The collateral cash flow data.

Example:

```
*   void *ptid = NULL;
*   CMO_STRUCT cmo;
*   memset(&cmo, 0, sizeof(CMO_STRUCT));
*   strcpy(cmo.dealid, "ACE06NC1");
*
*   open_deal_ex(ptid, &cmo);
*   run_deal_ex(ptid, &cmo);
*
*   // Get collateral flow of balance.
*      MARKIT_COLLAT_CASHFLOW MarkitCollatFlow;
*      int group_num = 1;
*   memset(&MarkitCollatFlow, 0, sizeof(MARKIT_COLLAT_CASHFLOW));
*   int ret = get_collateral_flow_ex1(ptid, group_num, NULL, &MarkitCollatFlow);
*   if(0 != ret)
*   {
*       const char* err_msg = get_deal_error_msg(ptid);
*       if(NULL != err_msg)
*       {
*           std::cout<< "Fail to call get_collateral_flow_ex1:" << err_msg << std::endl;
*       }
*   }
*
*      close_deal_ex(ptid, &cmo);
*
```

Note

User can call set_deal_calc_level() with parameter CALC_LEVEL_FULL after open_deal_ex() but before run-_deal_ex() to retrieve the following fields in MARKIT_COLLST_CASHFLOW: double gross_interest[MAX_-PERIODS], double sched_p_and_i[MAX_PERIODS], double negative_amortization[MAX_PERIODS], double draw_amount[MAX_PERIODS], double total_excess_losses[MAX_PERIODS], double po_balance[MAX-_PERIODS], double po_losses[MAX_PERIODS], double po_prepayment[MAX_PERIODS], double po_prin_-recoveries[MAX_PERIODS], double po_sched_principal[MAX_PERIODS], double premium_loan_balance[M-AX_PERIODS], double excess_interest[MAX_PERIODS].

See Also

- get_collateral_flow_ex() can return the specified group collateral cashflows. It needs the user to allocate the data structure MARKIT_CPLLAT_CASHFLOW.
- get_collateral_flow_ex1() is used to get specified collateral cashflows. It returns projected collateral cashflows specified by flow_identifier.

22.1.4.31 int CHASAPI get_collateral_id_ex (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int *loan_index,* const char ∗ *id_type,* char ∗ *id_array[ ],* int *id_array_length*)

Returns ID(s), such as CUSIP, for either all collateral or the requested piece of collateral in a deal

Since

2.1.1

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*loan_index*	The 0-based index of the loan (-1 for all loans).
in	*id_type*	Collateral id type. It is case-insensitive and should be: • "CUSIP"
in,out	*id_array*	A user allocated array to which IDs will be written.
in	*id_array_length*	The size of the id array. To make sure the API does not overrun user's memory.

Return values

>=0	Actual number of IDs returned
-1	Error - Deal not opened
-2	Error - Id type not support
-3	Error - Invalid loan index
-4	Error - Invalid output array size
-5	Error - Invalid output array
-6	Error - Actual pool data not loaded
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
     *      void* ptid=NULL;
     *      CMO_STRUCT deal={};
     *      strcpy(deal.dealid, "3437E");
     *      deal.actual_coll=1;
     *
     *      open_deal_ex(ptid, &deal);
     *      // deal is already open.
     *
     *      std::vector<char> id_buf(deal.num_colls*10);
     *      std::vector<char*> id(deal.num_colls);
     *      for(int i = 0; i<deal.num_colls; ++i)
     *      {
     *          id[i] = &id_buf[i*10];
     *      }
     *      int ret_val = get_collateral_id_ex(ptid, NULL, -1, "CUSIP", &id.front(), deal.
            num_colls);
     *      assert(ret_val > 0);
     *      // value of id[i] is the id of pool i
     *      // done
     *
     *      close_deal_ex(ptid, &deal);
     *
```

Note

If all collateral is requested, IDs array must be allocated to be at least as long as the value CMO_STRUCT.-num_colls returned by open_deal_ex().

22.1.4.32 int CHASAPI get_dates_from_upd_ex (void ∗ *tid,* char ∗ *szArchiveName,* int *UpdDate[]*)

Gets collateral update dates from an update file(∗.upd) and stores them in the output parameter "UpdDate". The return value of this method will be the number of dates that stores in "UpdDate".

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Return values

>=0	Number of dates returned
-1	Other error
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*szArchiveName*	Name of Moody's Analytics Deal file, ∗.chs or ∗.sfw. E.g. MA_Deal_File or MA_Deal_File.chs
out	*UpdDate*	Pointer to a client-allocated array of update dates (YYYYMM).

Note

The size of the output parameter must be greater than:

- The number of months between the current date and the deal settlement date, if which is smaller than MAX_PERIODS(==612).
- MAX_PERIODS (==612), if MAX_PERIODS is smaller than the number of months between the current date and the deal settlement date.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT cmo={};
*       strcpy(cmo.dealid, "AAM0401");
*
*       set_input_path("C:/deals");
*       set_engine_preference(
      PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
*       open_deal_ex(tid, &cmo);
*       // Deal is opened.
*
*       int UpdDate[MAX_PERIODS] = {0};
*       int nResults = get_dates_from_upd_ex(tid,cmo.dealid, UpdDate);
*       // nResults should be equal to 43.
*       // UpdDate[nResults] should be equal to 0.
*
*       close_deal_ex(tid, &cmo);
*
```

22.1.4.33 CALC_LEVEL CHASAPI get_deal_calc_level (void ∗ *tid*)

Gets the deal calculation level.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The dso identifier.

Return values

CALC_LEVEL	
CALC_LEVEL_BASIC	
CALC_LEVEL_FULL	

Example:

```
*     void * tid = NULL;
*     // Open deal
*
*     // Get deal calculation level
*     CALC_LEVEL calc_level = get_deal_calc_level(tid);
*
```

22.1.4.34 const char∗ CHASAPI get_deal_error_msg (void ∗ *tid*)

Retrieves text error message of the previous SDK call or NULL if no errors.

Since

0.9.0

Availability ALL

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls. After the function call, the parameter will be set to the error message or NULL.

Return values

0	No error
Address	Pointer to error message of the previous call

Example:

```
*   void* tid = NULL;
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   strcpy(pCmo->dealid,"AL2010-A");
*
*   int ret = open_deal_ex(tid, pCmo);
*   if (ret!=0)
*   {
*       const char * ptr_err_msg = get_deal_error_msg(tid);
*   }
*
*   close_deal_ex(tid, pCmo);
*   delete pCmo;
*   pCmo = NULL;
*
```

22.1.4.35 long CHASAPI get_deal_info (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* MARKIT_DEAL_INFO ∗ *deal_info*)

Populates the user allocated buffer with surveillance data for a specific month from the deal file, if the surveillance data is available. Please also refer to MARKIT_DEAL_INFO for details.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*deal_info*	Fields populated with deal information. It will be stored in MARKIT_DEAL_INFO.

Return values

0	Call succeeded - data has been copied to user_buffer and the length is actual_-size, the size of MARKIT_DEAL_INFO.
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0,sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Pre-condition: Deal is already opened
*
*       MARKIT_DEAL_INFO dealInfo;
*       memset(&dealInfo, 0, sizeof(MARKIT_DEAL_INFO));
*       int ret = get_deal_info(ptid, null, &dealInfo);
*       if(ret < 0)
*       {
*             std::cerr < "Error:" << get_deal_error_msg(ptid) << std::endl;
*        }
*
*       close_deal_ex(ptid,&deal);
*
```

22.1.4.36 int CHASAPI get_deal_issuer_type (void ∗ *tid,* char ∗ *Issuer,* char ∗ *Type*)

Retrieves the category of deal (Agency vs Whole_Loan) and type of deal within that category. Information is only retrieved if a non-null pointer is passed.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*Issuer*	A pointer to a character string used to return the deal type. The string must be at least 80 characters and allocated by the user. If NULL is passed, the variable Issuer will not be returned.
out	*Type*	A pointer to a character string used to return the deal category. The string must be at least 26 characters and allocated by the user. If NULL is passed, the variable Type will not be returned.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Check that output parameters are fully allocated
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFAC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened.
*
*       char issuer[80]={0};
*       char type[80]={0};
*       int iret=get_deal_issuer_type(ptid,issuer,type);
*
*       close_deal_ex(ptid,&deal);
*
```

22.1.4.37 int CHASAPI get_deal_payment_group (void ∗ *tid,* MARKIT_DEAL_PAYMENT_GROUP *group_array[ ],* int *group_array_size,* int ∗ *num_available*)

This method gets deal payment group information.

Since

2.6.0

Availability CHS

Precondition

open_deal_ex() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
`in,out`	*group_array*	The user allocated array to hold the group information.
`in`	*group_array_-size*	The size of the group_array.
`in,out`	*num_available*	Total number of available deal groups.

Return values

>=0	Success. Actual number of payment groups returned.
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* pDeal = NULL;
*       //Deal has been opened
*
*       MARKIT_DEAL_PAYMENT_GROUP payment_group_array[10];
*       memset(&payment_group_array, 0, sizeof(MARKIT_DEAL_PAYMENT_GROUP) * 10);
*       int ret = get_deal_payment_group(pDeal, payment_group_array, 10, &num_avail);
*
```

22.1.4.38 int CHASAPI get_deal_surv_data (void ∗ *tid,* MARKIT_DEAL_SURVEILLANCE_DATA ∗ *survData,* int *YYYYMM*)

This function retrieves the deal surveillance data as of the month and year provided in the format YYYYMM.

Since

1.1.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*YYYYMM*	the update month and year of the surveillance data for this deal
out	*survData*	the surveillance data structure, the memory is allocated by the caller

Return values

0	Success
<*0*	Error: for details call get_deal_error_msg()

Example:

```
*       void * tid = NULL;
*     // Open deal
*
*     // retrieves the deal surveillance data as of 201501
*     MARKIT_DEAL_SURVEILLANCE_DATA survData;
*     int rVal = get_deal_surv_data(tid, &survData,201501);
*
```

Deprecated This method is deprecated.

22.1.4.39 long CHASAPI get_group_info (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int *group_number,* MARKIT_GROUP_INFO ∗ *group_info*)

This retrieves the collateral group info given a group number.

Since

1.2.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	If reremic deal, this is the id, otherwise null

in	*group_number*	The group number
out	*group_info*	A pointer to the user allocated structure for group info.

Return values

0	Success
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*   void * tid = NULL;
*   // Deal is already opened
*
*   // get the total groups number
*   int numGroups(0);
*   view_coll_groups(tid, NULL, NULL, &numGroups);
*
*   //get group info
*   for(int i = 1; i<=numGroups; i++){
*       MARKIT_GROUP_INFO groupInfo;
*       get_group_info(tid, NULL,i,&groupInfo);
*   }
*
```

22.1.4.40 int CHASAPI get_hist_data_ex (void ∗ *tid*, CMO_STRUCT ∗ *cmos*, char ∗ *bondid*, double *hist_factor[ ]*, double *hist_coupon[ ]*)

This retrieves the bond's historical factors and coupons where available, in descending order by date, starting with the date the deal is opened "as of".

Since

1.1.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*cmos*	Pointer to the CMO_STRUCT used to open the deal.
in	*bondid*	The name of bond for which information is requested.
out	*hist_factor*	A pointer to a client-allocated vector of doubles which will return the historical factors. The vector must be large enough to hold MAX_PERIODS values.
out	*hist_coupon*	A pointer to a client-allocated vector of doubles which will return the historical coupons. The vector must be large enough to hold MAX_PERIODS values.

Return values

>=0	Number of periods
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*     void* tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT();
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "BAA03009");
*
*     set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*     assert(0 == open_deal_ex(tid, pCmo));
```

```
*
*      double hist_factor[MAX_PERIODS] = {0};
*      double hist_coupon[MAX_PERIODS] = {0};
*      char* bondid = "1CB1";
*      int ret = get_hist_data_ex(tid,pCmo,bondid,hist_factor,hist_coupon);
*      if(ret < 0)
*      {
*          //error handle
*      }
*
*      close_deal_ex(tid,pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

The latest information returned is for the settlement date specified when opening the deal. If no date was specified all available information is returned. Factors and coupons of -1 indicate that information is not available for that date.

22.1.4.41 int CHASAPI get_hist_data_ex1 (void ∗ *tid*, CMO_STRUCT ∗ *cmos*, char ∗ *bondid*, int *date[ ]*, double *principal_losses[ ]*, double *paid_interest[ ]*)

This retrieves the bond's historical principal losses and paid interest where available, in descending order by date, starting with the date the deal is opened "as of".

Since

1.3.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*cmos*	Pointer to the CMO_STRUCT used to open the deal.
in	*bondid*	The name of bond for which information is requested.
out	*date*	A pointer to a client-allocated vector of integers which will return the historical date. It is in descending order from the settlement date specified. The output format is yyyymm.
out	*principal_losses*	A pointer to a client-allocated vector of doubles which will return the historical principal losses. The vector must be large enough to hold MAX_PERIODS values.
out	*paid_interest*	A pointer to a client-allocated vector of doubles which will return the historical paid interest. The vector must be large enough to hold MAX_PERIODS values.

Return values

>=0	Number of periods
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Note

The latest information returned is for the settlement date specified when opening the deal. If no date was specified all available information is returned. For CHS deals, values of -1 indicate that information is not available for that date; for SFW deals, values of -9.123E+19 indicate that information is not available for that date. For CHS deals, paid_interest is calculated from hist_coupon and hist_factor, and hist_principal loss is always 0.

Example:

```
*        void* tid = NULL;
*        CMO_STRUCT *pCmo = new CMO_STRUCT();
*        memset(pCmo, 0, sizeof(*pCmo));
*        strcpy(pCmo->dealid, "AL2010-A");
*
*        set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*        assert(0 == open_deal_ex(tid, pCmo));
*
*        int hist_date[MAX_PERIODS] = {0};
*        double hist_prinlosses[MAX_PERIODS] = {0};
*        double hist_paidint[MAX_PERIODS] = {0};
*        char* bondid = "A";
*        int ret = get_hist_data_ex1(tid,pCmo,bondid,hist_date,hist_prinlosses,hist_paidint
      );
*        if(ret < 0)
*        {
*            //error handle
*        }
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.42 const char∗ CHASAPI get_input_path ( )

Returns the path to the SDK Deals. The path is stored in the static variable, input_subdiretory. The initial value of this variable is "\n" and can be reset by set_input_path().

Since

0.9.0

Availability ALL

Precondition

set_input_path() has been called.

Returns

The path to the SDK deals.

Example:

```
*       set_input_path("C:/Test");
*       const char *path = get_input_path();
*
```

Note

The input_path is initials with "\n". If the user don't call set_input_path() function, get_input_path() will return "\n".

22.1.4.43 int CHASAPI get_license_info (int *num_features,* LICENSE_INFO *lic_info[]*)

Retrieves the license information.

Since

3.0.0

Availability SFW, CDOnet, CHS

Parameters

in	*num_features*	The number of license features.
out	*lic_info*	License informations.

Return values

>=0	Number of license features
-1	Error - WSA_API.LIC not found
-99	Error: Other errors, please see details by calling get_deal_error_msg(NULL)

Example:

```
     *      int arraySize = get_license_info(0, NULL);
     *      LICENSE_INFO* pLisenceInfo = new LICENSE_INFO[arraySize];
     *      memset(pLisenceInfo, 0, sizeof(LICENSE_INFO)*arraySize);
     *      int iRet = get_license_info(arraySize, pLisenceInfo);
     *      delete [] pLisenceInfo;
     *
```

22.1.4.44 int CHASAPI get_loan_level_avail_YYYYMMs (void ∗ *tid,* int *YYYYMMs[ ],* int *sizeOfYYYYMMs,* int ∗ *numAvailable*)

Returns the available dates for loan level data, the format is in YYYYMM.

Since

1.2.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*sizeOfYYYYM-Ms*	The size of array provided
in,out	*YYYYMMs*	List of available loan level data dates.
in,out	*numAvailable*	Total number of available YYYYMMs.

Returns

actual number of dates returned to the caller.

Example:

```
     *      void* tid = NULL;
     *      CMO_STRUCT *pCmo = new CMO_STRUCT();
     *      memset(pCmo, 0, sizeof(*pCmo));
     *      strcpy(pCmo->dealid, "AB05HE3");
     *
     *      set_engine_preference(
           PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
     *      assert(0 == open_deal_ex(tid, pCmo));
     *
     *      int expected_dates = 30;
     *      int dates[30];
     *      int numAvailable=0;
     *      int files=0;
     *
     *      files = get_loan_level_avail_YYYYMMs(tid, dates, expected_dates, &
           numAvailable);
     *
     *      if(files < 0)
     *      {
     *          //error handle
     *      }
     *
     *      assert(0 == close_deal_ex(tid, pCmo));
     *      delete pCmo;
     *      pCmo = NULL;
     *
```

22.1.4.45 long CHASAPI get_longest_ex (void ∗ *tid*)

Returns the number of remain periods to maturity of collateral. This can be used to increase efficiency by only processing periods where there may be activity (both in setting scenarios and processing cashflows).

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

>=0	Number of periods
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*      set_input_path("C:/Deals");
*      set_engine_preference(
      PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
*
*      void* tid=NULL;
*      CMO_STRUCT cmo={};
*      strcpy(cmo.dealid, "AAM0401");
*      open_deal_ex(tid, &cmo);
*      // Deal is already opened
*
*      int longest=get_longest_ex(tid);
*      char cf_date[11]={};
*      for (int i=0; i<=longest; ++i)
*      {
*          char* msg=get_cf_date(i, cf_date, tid);
*          std::cout << msg << std::endl;
*      }
*
*      close_deal_ex(tid, &cmo);
*
```

22.1.4.46 int CHASAPI get_markit_bond_pool_history (void ∗ *tid*, const char ∗ *cusip*, const int *history_identifier*, MARKIT_POOL_HISTORY_DATA *pool_history[ ]*, int *size_array*, int *YYYYMM*)

This method gets the historical data item of YYYYMM for the specified cusip, if YYYYMM=0 would return all available historical data for the specified bond cusip.

Since

2.4.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*cusip*	The CUSIP of the bond. if cusip equals NULL, return with deal level history.
in	*history_identifier*	Identifies of the history data. Must be one of the markit pool history identifiers (see MARKIT_POOL_HISTORY).
in	*size_array*	The size of array pool_history provided.
in	*YYYYMM*	The specified date (of format YYYYMM),if YYYYMM=0 would return all historical data for the specified cusip.
out	*pool_history*	The list of pool history. This parameter must be pre-allocated before call this function.

Return values

>0	Actual size of history array returned.
0	None history data.
-99	Error ,call get_deal_error_msg() for details.

Example:

```
 *      void* pDeal = NULL;
 *      //Deal has been opened
 *
 *      const char* cusip = "38375PKW0";
 *      MARKIT_POOL_HISTORY_DATA  poolhist[1] = {0};
 *      int ret = get_markit_bond_pool_history(pDeal, cusip,
       MARKIT_POOL_HISTORY_CPR1M, poolhist, 1, 201507);
 *
```

22.1.4.47 int CHASAPI get_markit_bond_pool_history_avail_YYYYMMs (void ∗ *tid,* const char ∗ *cusip,* int *YYYYMMs[ ],* int *size_YYYYMMs,* int ∗ *num_available*)

Returns all available history dates for the specified bond, the format is in YYYYMM.

Since

2.4.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*cusip*	The CUSIP of the bond. if cusip equals NULL, return with deal level history available YYYYMMs.
in	*size_YYYYMMs*	The size of array provided
in,out	*YYYYMMs*	List of available history data dates.
in,out	*num_available*	Total number of available YYYYMMs.

Return values

>0	Actual number of dates returned to the caller.
0	None available history dates.
-99	Error ,call get_deal_error_msg() for details.

Example:

```
 *      void* pDeal = NULL;
 *      //Deal has been opened
 *
 *      const char* cusip = "38375PKW0";
 *      int yyyymms[1] = {0};
```

```
    *       int num_avail = 0;
    *       int ret = get_markit_bond_pool_history_avail_YYYYMMs(pDeal
          ,cusip, yyyymms, 1, &num_avail);
    *
```

22.1.4.48 int CHASAPI get_markit_id (const char * *id*, char * *deal*, char * *bond*)

Returns the SDK deal and bond ID for an industry-standard bond identifier.

Since

0.9.0

Availability ALL

Parameters

in	*id*	The US or international bond identifier.
out	*deal*	The deal ID. At least 13 characters must have been allocated.
out	*bond*	The bond ID. At least 7 characters must have been allocated.

Return values

-2	Error - Invalid identifier
-1	Error - File missing
0	Identifier not found
1	Identifier found in US securities file
2	Identifier found in international securities file

See Also

get_moodys_id()

Deprecated This method is deprecated, use get_moodys_id() instead.

22.1.4.49 int CHASAPI get_markit_id1 (const char * *id*, char * *deal*, char * *bond*, char * *err_buffer*, int *err_length*)

Returns the SDK deal and bond ID for an industry-standard bond identifier. Comparing with get_markit_id(), this function reports error messages through its last 2 parameters.

Since

0.9.0

Availability ALL

Parameters

in	*id*	The US or international bond identifier.
out	*deal*	The deal ID. At least 13 characters must have been allocated.
out	*bond*	The bond ID. At least 7 characters must have been allocated.
out	*err_buffer*	The error message.

in	*err_length*	The length of the error message.

Return values

-99	Please examine err_buffer for error
0	No error, but deal is not found
1	Found deal successfully

See Also

get_moodys_id()

Deprecated This method is deprecated, use get_moodys_id() instead.

22.1.4.50 int CHASAPI get_markit_pool_history (const char * *cusip*, const int *history_identifier*, MARKIT_POOL_HISTORY_DATA *pool_history[ ]*, int *size_array*, int *YYYYMM*)

This method gets the historical data item of YYYYMM for the specified cusip, if YYYYMM=0 would return all available historical data for the specified cusip.

Since

2.4.0

Availability CHS

Parameters

in	*cusip*	The CUSIP of the pool.
in	*history_identifier*	Identifies of the history data. Must be one of the markit pool history identifiers (see MARKIT_POOL_HISTORY).
in	*size_array*	The size of array pool_history provided.
in	*YYYYMM*	The specified date (of format YYYYMM),if YYYYMM=0 would return all historical data for the specified cusip.
out	*pool_history*	The list of pool history. This parameter must be pre-allocated before call this function.

Return values

$>$*0*	Actual size of history array returned.
0	None history data.
-99	Error ,call get_deal_error_msg() for details.

Example:

```
*       const char* cusip = "31295WXK9";
*       MARKIT_POOL_HISTORY_DATA  poolhist[1] = {0};
*       int ret = get_markit_pool_history(cusip,
      MARKIT_POOL_HISTORY_CPR1M, poolhist, 1, 201507);
*
```

22.1.4.51 int CHASAPI get_markit_pool_history_avail_YYYYMMs (const char * *cusip*, int *YYYYMMs[ ]*, int *size_YYYYMMs*, int * *num_available*)

Returns all available history dates for the specified pool, the format is in YYYYMM.

Since

2.4.0

Availability CHS

Parameters

in	*cusip*	The CUSIP of the pool.
in	*size_YYYYMMs*	The size of array provided
in,out	*YYYYMMs*	List of available history data dates.
in,out	*num_available*	Total number of available YYYYMMs.

Return values

>0	Actual number of dates returned to the caller.
0	None available history dates
-99	Error ,call get_deal_error_msg() for details.

Example:

```
*       const char* cusip = "3128S3EF2";
*       int yyyymms[1] = {0};
*       int num_avail = 0;
*       int ret = get_markit_pool_history_avail_YYYYMMs(cusip, yyyymms,
        1, &num_avail);
*
```

22.1.4.52 MARKIT_POOL_INFO∗ CHASAPI get_next_collat (void ∗ *tid,* void ∗ *collat_iterator*)

This function gets next collateral information using iterator obtained from calling obtain_collat_iterator(). When iterator goes to the end of the collateral set, it returns NULL.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

obtain_collat_iterator() has been called.

Parameters

in	*tid*	The deal/scenario object identifier
in	*collat_iterator*	Pointer to collateral iterator returned by calling obtain_collat_iterator().

Return values

Pointer	Pointer to the next collateral information
0	No more information left

Example:

```
*       void * tid = NULL;
*       // deal has been opened
*
*       MARKIT_POOL_INFO* coll_info =0;
*       void* coll_it =obtain_collat_iterator(tid, 0);
*       if(coll_it == 0)
*       {
*           std::cout << "Failure to start collat iteration " <<
       get_deal_error_msg(tid) << std::endl;
*       }
*       while(coll_info =  get_next_collat(tid,coll_it))
*       {
*           // do what you need with collateral
*       }
*
```

Note

Function returns pointers to collateral information allocated by the API. These pointers will be valid until deal is closed or another call to obtain_collat_iterator() function is made using the same parameters. The iterator will be released when close_deal_ex() is called. The iterator will be overwritten when obtain_collat_iterator is called again.

See Also

get_next_collat_for_managed_code

22.1.4.53 int CHASAPI get_next_collat_for_managed_code (void ∗ *tid*, void ∗ *collat_iterator*, MARKIT_POOL_INFO ∗ *usr_pool*, CCMO_ARM_INFO ∗ *arm*, MARKIT_PAYMENT_SCHEDULE ∗ *sched*, MARKIT_PREPAY_PENALTY *prepayPenalty[ ]*, int *sizeOfPpenArray*, int ∗ *hasArm*, int ∗ *hasSched*, int ∗ *hasPpen*)

This function is for managed code and gets the next collateral information using iterator obtained from calling obtain-_collat_iterator(). All structures should have been allocated by the caller. There is no memory allocated by SDK.

Since

1.0.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*collat_iterator*	Pointer to collateral iterator returned by calling obtain_collat_iterator().
in	*sizeOfPpenArray*	Periods of prepayment penalty data requested.
out	*usr_pool*	The user allocated buffer to hold pool data.
out	*arm*	The user allocated buffer to hold adjustable rate information of usr_pool.
out	*sched*	The user allocated buffer to hold payment schedule data of usr_pool.
out	*prepayPenalty*	A pointer to a client-allocated array of MARKIT_PREPAY_PENALT in which the prepayment penalty data of usr_pool will be stored.
out	*hasArm*	Set to 1 if arm is set.
out	*hasSched*	Set to 1 if sched is set.
out	*hasPpen*	Just set to 0.

Return values

1	End of collateral list.
0	Collateral is successfully loaded.
-1	collat_iterator is NULL.
-2	usr_pool should be allocated by caller
-3	arm should be allocated by caller
-4	sched should be allocated by caller
-6	hasArm should be allocated by caller
-7	hasSched should be allocated by caller
-8	hasPpen should be allocated by caller
-99	Error, Invalid dso identifier (tid) or other errors, please see details by calling get_-deal_error_msg()

Example:

```
*       void * tid = NULL;
*       //Deal has opened
*
```

```
*   MARKIT_POOL_INFO pool;
*   CCMO_ARM_INFO arm;
*   MARKIT_PAYMENT_SCHEDULE shed;
*   MARKIT_PREPAY_PENALTY prepayPenalty[10];
*   int has_Arm;
*   int has_Sched;
*   int hasPpen;
*   int coll_info;
*
*   void*coll_it=obtain_collat_iterator(tid,0);
*   if(coll_it==0)
*   {
*       std::cout<<"Failure to start collat iteration"<<get_deal_error_msg(tid_<<
      std::endl;
*          return;
*   }
*   int iret=get_next_collat_for_managed_code(tid,coll_it,&pool,&arm,&sched
      ,&prepayPenalty,10,&has_Arm,&has_Sched,&hasPpen);
*      if (0==iret)
*   {
*       //do what you want with pool
*       If(has_Arm)
*       {
*           //do what you want with arm
*       }
*       If(has_Sched)
*       {
*           //do what you want with sched
*       }
*       If(has_Ppen)
*       {
*           //do what you want with prepayPenalty
*       }
*   }
*      else if (iret<0)
*      {
*          //Error handling
*      }
*
```

See Also

get_next_collat()

22.1.4.54 void CHASAPI get_pool_by_index_ex (void ∗ *tid*, CCMO_POOL_INFO ∗ *p*, long *index*)

Gets collateral information for the specified piece of collateral.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*p*	A pointer to a client-allocated CCMO_POOL_INFO structure.
in	*index*	The 0-based index of the piece of collateral.

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "STATICLO");
*
*      set_engine_preference(
     PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(tid, pCmo));
*
*      CCMO_POOL_INFO poolInfo;
*      memset(&poolInfo, 0, sizeof(CCMO_POOL_INFO));
*       get_pool_by_index_ex(tid, &poolInfo, 7);
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.55 CCMO_POOL_INFO∗ CHASAPI get_pool_ptr_by_index_ex (void ∗ *tid*, long *index*)

Returns a pointer to the collateral specified by the index. This can be used to modify collateral characteristics.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*index*	The 0-based index of the piece of collateral.

Return values

NULL	Invalid collateral index
OTHER	Pointer to the CCMO_POOL_INFO structure for that piece of collateral

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       CCMO_POOL_INFO *pPoolInfo = NULL;
*       pPoolInfo = get_pool_ptr_by_index_ex(tid, 0);
*       if(pPoolInfo == NULL)
*       {
*
*       }
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.56 double∗ CHASAPI get_rate_ex (void ∗ *tid*, short *index*)

Gets the rate array for the given index.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
`in`	*index*	The rate index (enum of INDEX_TYPE or INDEX_TYPE_EX).

Return values

NULL	Error, please see details by calling get_deal_error_msg().
Other	The array pointer to the index rate, the array length is MAX_PERIODS(=612).

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened.
*
*       double *pfRate=NULL;
*       pfRate=get_rate_ex(ptid, LIBOR_3);
*
*       close_deal_ex(ptid,&deal);
*
```

See Also

get_required_rate_codes()

22.1.4.57 long CHASAPI get_rates_ex (void * *tid*, short * *ps_rates*)

Returns number of interest rate indices used by current deal and populates ps_rates array to indicate the interest rate indices used.

ps_rates array might be following:

Array Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**...**
Element Value	1	0	1	0	0	0	0	...
Mapping to Index rates	LIBOR_1	LIBOR_3	LIBOR_6	LIBOR_-12	LIBOR_-24	LIBOR_-36	LIBOR_-48	...

The ps_rates above suggesting , the current deal is using Libor 1 month and Libor 6 month index rate. The mapping comes from INDEX_TYPE and INDEX_TYPE_EX

Deprecated This method is deprecated. Use get_required_rate_codes().

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*ps_rates*	The user allocated array (size>=MAX_INDEX_TYPES_EX) to indicate interest rate indices used by current deal. • 1 - The interest rate is used • 0 - The interest rate is not used

Return values

>=0	The number of rates used
-1	Deal not open
-2	Invalid rates pointer
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already open
*
*       // Determine which market rates are required and set them
*       short mktRates[MAX_INDEX_TYPES_EX] = {0};
*       get_rates_ex(ptid, mktRates);
*
*       // Set the market rate for LIBOR_1 as a constant if it is required
*       if(1 == mktRates[LIBOR_1])
*       {
*           type = LIBOR_1;
*           rate = .0525;                       //5.25%
*           set_rate_ex(ptid, &type, 0, &rate);
*       }
*
*       // Set the market rate for LIBOR_6 as a vector if it is required
*       // rateVec is a vector containing the market rates
*       // rateVecSize is the number of rates in the vector to use
*       if(1 == mktRates[LIBOR_6])
*       {
*          type = LIBOR_6;
*          set_rate_ex(ptid, &type, rateVecSize, rateVec);
*       }
*
*       close_deal_ex(ptid,&deal);
*
```

See Also

get_rate_ex() and get_required_rate_codes()

22.1.4.58 int CHASAPI get_repline_index_list (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, int *loan_index*, int *repline_array[ ]*, int *repline_array_length*)

Returns repline pool index for either all collateral or the requested piece of collateral in a deal

repline_array element can be:

- >=0, repline pool index
- -1, no matching repline pool

Since

2.1.1

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*loan_index*	The 0-based index of the loan (-1 for all loans).
in,out	*repline_array*	A user allocated array to which repline index will be written.
in	*repline_array_-length*	The size of the repline_array. To make sure the API does not overrun user's memory.

Return values

>=0	Actual number of repline indices returned
-1	Error - Deal not opened
-3	Error - Invalid loan index
-4	Error - Invalid output array size
-5	Error - Invalid output array
-6	Error - Actual pool data not loaded
-7	Error - Repline info not available/loaded
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal={};
*       strcpy(deal.dealid, "3437E");
*       deal.actual_coll=1;
*
*       open_deal_ex(ptid, &deal);
*       // deal is already open.
*
*       int repline_index[10]={};
*       int ret_val = get_repline_index_list(ptid, NULL, -1, repline_index, 10);
*       assert(ret_val > 0);
*       // value of repline_index[i] is the repline pool index of pool i
*       // done
*
*       close_deal_ex(ptid, &deal);
*
```

Note

If all collateral is requested, repline_array must be allocated to be at least as long as the value CMO_STRU-CT.num_colls returned by open_deal_ex().

22.1.4.59 int CHASAPI get_required_rate_codes (void ∗ *tid,* int ∗ *rate_codes,* int *size_of_array_codes*)

Returns number of interest rate indices used by current deal and populates rate_codes array with the list of index rate codes that are used.

one example rate_codes array might be following:

Array Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**...**

Element Value	1	3	0	0	0	0	0	...

The element value suggests the enum value from INDEX_TYPE and INDEX_TYPE_EX.

In this case, 1 maps to LIBOR_1 and 3 maps to LIBOR_6, which means ,current deal use Libor 1 month and Libor 6 month rates.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*rate_codes*	The list of index rates codes used in the deal.
in	*size_of_array_-codes*	The size of the user allocated array rate_codes.

Return values

>=0	The number of rates used
-1	Deal not open
-2	Invalid rates pointer
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid, "BAFC08R2");
*
*       open_deal_ex(ptid, &deal);
*       // Deal is already open.
*
*       int required_rates[MAX_INDEX_TYPES_EX]={};
*       // the codes of index rates will return in 'required_rates'.
*       int actual_count = get_required_rate_codes(ptid, required_rates,
       MAX_INDEX_TYPES_EX);
*       assert(actual_count > 0);
*       if (LIBOR_12 == required_rates[0])
*       {
*           // do something
*       }
*       else
*       {
*           // ...
*       }
*
*       close_deal_ex(ptid, &deal);
*
```

Note

This method is the replacement for deprecated get_rates_ex().

See Also

get_rate_ex()

22.1.4.60 int CHASAPI get_reremic_bond_band (void ∗ *tid*, char ∗ *dealid*, const char ∗ *bondid*, double ∗ *pricing_wal*, double ∗ *low*, double ∗ *high*)

Retrieves the band information for the bond in the specified underlying deal (pricing WAL, low speed and high speed). If NULL is passed for any item, that item will not be returned.

Since

1.1.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*bondid*	A pointer to the name of the bond.
out	*pricing_wal*	The weighted average life of the bond when the deal is priced.
out	*low*	The low collar speed for the band.
out	*high*	The high collar speed for the band.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-10	Error - Bond not found
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF04OPT4N");
*
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       double pricing_wal = 0.0;
*       double low = 0.0;
*       double high = 0.0;
*       int remic_num = view_reremic_deals(tid,NULL,NULL);
*       if(remic_num>0)
*       {
*           CMO_STRUCT *remics = (CMO_STRUCT*)malloc(remic_num * sizeof(
      CMO_STRUCT));
*           remic_num = view_reremic_deals(tid,NULL,remics);
*           assert(0 == get_reremic_bond_band(tid, remics[0].dealid, "A1", &pricing_wal
      , &low, &high));
*       }
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.61 int CHASAPI get_reremic_bond_misc (void ∗ *tid*, char ∗ *dealid*, const char ∗ *Bond*, BOOLYAN ∗ *IsSeg*, BOOLYAN ∗ *IsMACR*, BOOLYAN ∗ *IsPO*)

Get additional information on a bond (segment, MACR or PO) in the underlying deal.

Since

1.2.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*Bond*	A pointer to the name of the bond.
out	*IsSeg*	If non-zero, the bond is a segment bond. Otherwise it is not.
out	*IsMACR*	If non-zero, the bond is a MACR bond. Otherwise it is not.
out	*IsPO*	If non-zero, the bond is a PO bond. Otherwise it is not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-10	Error - Bond not found
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*     void* ptid = NULL;
*     // deal has been opened
*
*     int remic_num = view_reremic_deals(ptid,NULL,NULL);
*     if(remic_num>0)
*     {
*         std::vector<CMO_STRUCT>remics(remic_num);
*         remic_num = view_reremic_deals(ptid,NULL,&remics.front());
*         BOOLYAN isSeg = 0;
*         BOOLYAN isMacr = 0;
*         BOOLYAN isPo = 0;
*         for(int i = 0;i<remic_num;++i)
*         {
*             int iret = get_reremic_bond_misc(ptid, remics[0].dealid, remics[0].bond.
      stripped_id, &isSeg, &isMacr, &isPo);
*             if (iret < 0)
*             {
*                 //error handle
*             }
*         }
*     }
*
```

See Also

- get_bond_misc_ex()

22.1.4.62 CCMO_POOL_INFO∗ CHASAPI get_reremic_pool_ptr_by_index (void ∗ *tid,* char ∗ *dealid,* long *index,* int & *error*)

Returns a pointer to the collateral in the underlying deal specified by the index. This can be used to modify collateral characteristics.

Since

1.3.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*index*	The 0-based index of the piece of collateral.
out	*error*	Error number. Values are: • -1 Deal not opened • -2 Requested underlying deal not part of the deal • -3 Invalid collateral index. • -99 Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Return values

NULL	Error: Reason returned in error parameter
OTHER	Pointer to CCMO_POOL_INFO structure for requested collateral

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF04OPT4N");
*
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       int poolIdx = 1;  // The 0-based index of the piece of collateral
*       CCMO_POOL_INFO *pPoolInfo = NULL;
*       int error = 0;
*       int remic_num = view_reremic_deals(tid, NULL, NULL);
*       if(remic_num > 0)
*       {
*           CMO_STRUCT *remics = (CMO_STRUCT*)malloc(remic_num * sizeof(
      CMO_STRUCT));
*           remic_num = view_reremic_deals(tid, NULL, remics);
*           pPoolInfo = get_reremic_pool_ptr_by_index(tid, remics[0].dealid,
      poolIdx, error);
*           if(pPoolInfo == NULL)
*           {
*               // Invalid collateral index
*           }
*       }
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.63 int CHASAPI get_reremic_trigger_status (void * *tid,* char * *dealid,* char * *trigger_name,* SBYTE * *status*)

Returns the status information for the requested trigger in the underlying deal.

Since

1.2.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*trigger_name*	The case-sensitive name of the trigger that status information is requested for.
out	*status*	A pointer to a client-allocated array of SBYTE(signed char). Allocate MAX_P-ERIODS.

Return values

0	No Error
-1	Error Deal not opened
-2	Error Requested underlying deal not part of the deal
-3	Error Trigger not in deal
-99	Error Invalid dso identifier (tid) or other errors, please see details by calling get_-deal_error_msg()

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF04OPT4N");
*
*      open_deal_ex(tid, pCmo);
*
*      signed char tmp_status[MAX_PERIODS];
*      int remic_num = view_reremic_deals(tid, NULL, NULL);
*      if(remic_num > 0)
*      {
*          CMO_STRUCT *remics = (CMO_STRUCT*)malloc(remic_num * sizeof(
     CMO_STRUCT));
*          remic_num = view_reremic_deals(tid,NULL,remics);
*          int trg_num = get_reremic_triggers_avail(tid, remics[0].dealid, NULL,
     NULL);
*
*          if(trg_num > 0)
*          {
*              char *name_buf = (char*)malloc(trg_num*21);
*              char **names = (char**)malloc(trg_num * sizeof(char*));
*              for(int i = 0; i<trg_num; ++i)
*              {
*                  names[i] = (char*)malloc(21);
*              }
*              signed char *status_buf = (signed char*)malloc(trg_num*MAX_PERIODS);
*              signed char **status = (signed char**)malloc(trg_num * sizeof(signed char*));
*              for(int i = 0; i<trg_num; ++i)
*              {
*                  status[i] = (signed char*)malloc(MAX_PERIODS);
*              }
*              for(int i = 0; i<trg_num; i++ )
*              {
*                  names[i] = &name_buf[i*21];
*                  status[i] = &status_buf[i*MAX_PERIODS];
*              }
*              get_reremic_triggers_avail(tid, remics[0].dealid, names, NULL);
*
*              for(int i = 0; i<trg_num; i++)
*              {
*                  get_reremic_trigger_status(tid, remics[0].dealid, names[i],
     status[i]);
*              }
*          }
*      }
*
*      close_deal_ex(tid,pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.1.4.64 int CHASAPI get_reremic_triggers_avail (void ∗ *tid,* char ∗ *dealid,* char ∗ *trigger_names[ ],* char ∗ *trigger_descs[ ]*)

This retrieves the names and/or descriptions of the triggers in the underlying deal.

Since

1.2.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
out	*trigger_names*	A pointer to a client-allocated array of character strings in which the names of the triggers will be stored. 21 characters should be allocated for each string.
out	*trigger_descs*	A pointer to a client-allocated array of character strings in which the descriptions of the triggers will be stored. 1025 characters should be allocated for each string. Pass NULL if descriptions are not required.

Return values

>=0	Number of miscellaneous variables
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Note

Triggers are conditions that affect the custom paydown rules. In each period the trigger will either fail (condition no met – no action) or pass (action taken). The value(s) of one of these variables can be obtained by calling get_trigger_status() after running the deal (run_deal_ex()). The triggers can be overridden by calling set_-trigger_override() before calling run_deal_ex().

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF04OPT4N");
*
*       open_deal_ex(tid, pCmo);
*
*       signed char tmp_status[MAX_PERIODS];
*       int remic_num = view_reremic_deals(tid, NULL, NULL);
*       if(remic_num > 0)
*       {
*           CMO_STRUCT *remics = (CMO_STRUCT*)malloc(remic_num * sizeof(
       CMO_STRUCT));
*           remic_num = view_reremic_deals(tid,NULL,remics);
*
*           int trg_num = get_reremic_triggers_avail(tid, remics[0].dealid, NULL,
       NULL);
*           if(trg_num < 0)
*           {
*               //error handling
*           }
*       }
*
*        close_deal_ex(tid,pCmo);
*        delete pCmo;
*        pCmo = NULL;
*
```

22.1.4.65 const char∗ CHASAPI get_sdk_build_version ( )

This function returns the Software release version of this WSA API build.

Since

1.1.0

Availability ALL

Return values

0	Error
Other	The version number of the build

Example:

```
*       const char* version = get_sdk_build_version();
*       bool support_exess_rate = strcmp(version,"2, 4, 1, 0") > 0;
*
```

22.1.4.66 int CHASAPI get_surv_avail_YYYYMMs (void * *tid*, int *YYYYMMs[ ]*, int *sizeOfYYYYMMs*, int * *numAvailable*)

This function retrieves the deal surveillance data as of the month and year provided in the format YYYYMM. The returned YYYYMMs are sorted in descending order, i.e., from the latest to oldest in time. If the array YYYYMMs is not big enough to hold all available data, only the latest ones will be filled in the array. The total available is passed back to user through *numAvailable. The user can re-allocate an array based on this value and call the function again.

Since

1.0.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*sizeOfYYYYMMs*	The size of array YYYYMMs passed in
out	*YYYYMMs*	Array to hold the available year and month
out	*numAvailable*	Total number of available surveillance data YYYYMMs

Returns

The number of YYYYMMs passed back in the YYYYMMs array.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "CW05J7");
*
*       open_deal_ex(tid, pCmo);
*
*       int expected_dates = 50;
*       int *surv_dates = (int*)malloc(expected_dates * sizeof(int));
*       memset(surv_dates, 0, sizeof(*surv_dates)*expected_dates);
*       int numAvailable=0;
*       int survFiles = get_surv_avail_YYYYMMs(tid, surv_dates, expected_dates, &
      numAvailable);
*       if(survFiles < 0)
*       {
*        //error handle
*       }
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.67 int CHASAPI get_surveillance_data (void ∗ *tid,* int *YYYYMM,* char ∗ *user_buffer,* long *size_of_user_buffer,* long ∗ *actual_size*)

Populates the user allocated buffer with surveillance data for a specific month, if the surveillance data is available.

Since

1.0.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*YYYYMM*	The year and month when the surveillance data was collected.
in,out	*user_buffer*	The user allocated buffer to hold output data.
in	*size_of_user_-buffer*	The size of the allocated buffer.
out	*actual_size*	Actual size of the surveillance data.

Return values

0	Call succeeded, and data has been copied to user_buffer, and the length is actual-_size.
-1	Error - Deal not opened
-2	Error - Surveillance data is not available for the specified date.
-3	Error - User allocated buffer is not big enough to hold the data. Should call again with a bigger buffer, at least of size actual_size.
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void * tid = NULL;
*       // Pre-condition: Deal is already opened
*
*       int YYYYMM = 200804;
*       const int SURV_DATA_SIZE_SMALL = 2048;
*       const int SURV_DATA_SIZE_BIG   = 4096;
*       char surv_buf_s[SURV_DATA_SIZE_SMALL];
*       char surv_buf_b[SURV_DATA_SIZE_BIG];
*
*       size_t actSize=0;
*       int ret = get_surveillance_data(tid, YYYYMM, surv_buf_s,
*           SURV_DATA_SIZE_SMALL,&actSize);
*       if(ret < 0)
*       {
*           if (ret == -3)
*           {
*               std::cout << "Error: Buffer size is "
*               << SURV_DATA_SIZE_SMALL
*               << ". Need " << actual_size << "."
*               << std::endl;
*               ret = get_surveillance_data(tid, YYYYMM, surv_buf_b,
*                   SURV_DATA_SIZE_BIG,&actSize);
*               if (ret >= 0)
*               {
*                   std::cout << surv_buf << std::endl;
*               }
*           }
*       }
*       else
*       {
*           std::cout << surv_buf << std::endl;
*       }
*
```

22.1.4.68 int CHASAPI get_trigger_status (void ∗ *tid,* char ∗ *trigger_name,* SBYTE ∗ *status*)

Returns the status information for the requested trigger.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*trigger_name*	The case-sensitive name of the trigger that status information is requested for.
out	*status*	A pointer to a client-allocated array of SBYTE(signed char). Allocate MAX_PERIODS.

Return values

0	No Error
-1	Error - Deal not opened
-3	Error - Trigger not in deal
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AB05HE3");
*
*       set_engine_preference(
      PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
*       open_deal_ex(tid, pCmo);
*
*       int trg_num = get_triggers_avail(tid, NULL, NULL);
*       if(trg_num>0)
*       {
*           char *name_buf = (char*)malloc(trg_num*21);
*           char **names = (char**)malloc(trg_num * sizeof(char*));
*           for(int i = 0; i<trg_num; ++i)
*           {
*               names[i] = (char*)malloc(21);
*           }
*           signed char *status_buf = (signed char*)malloc(trg_num*MAX_PERIODS);
*           signed char **status = (signed char**)malloc(trg_num * sizeof(signed char*));
*           for(int i = 0; i<trg_num; ++i)
*           {
*               status[i] = (signed char*)malloc(MAX_PERIODS);
*           }
*           for(int i = 0; i<trg_num; i++ )
*           {
*               names[i] = &name_buf[i*21];
*               status[i] = &status_buf[i*MAX_PERIODS];
*           }
*
*           get_triggers_avail(tid, names, NULL);
*           for(int i = 0; i<trg_num; i++)
*           {
*               get_trigger_status(tid, names[i], status[i]);
*           }
*       }
*
*       close_deal_ex(tid, pCmo);
*
```

22.1.4.69 int CHASAPI get_triggers_avail (void ∗ *tid,* char ∗ *trigger_names[],* char ∗ *trigger_descs[]*)

This retrieves the names and/or descriptions of the triggers in the deal.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*trigger_names*	A pointer to a client-allocated array of character strings in which the names of the triggers will be stored. 21 characters should be allocated for each string. Only filled in if it is not NULL.
out	*trigger_descs*	A pointer to a client-allocated array of character strings in which the descriptions of the triggers will be stored. 1025 characters should be allocated for each string. Pass NULL if descriptions are not required. Only filled in if not NULL.

Return values

>=0	Number of triggers
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Note

Pass NULL for trigger_names and trigger_descs to get just the number of triggers. Triggers are conditions that affect the custom paydown rules. In each period the trigger will either fail (condition no met – no action) or pass (action taken). The value(s) of one of these variables can be obtained by calling get_trigger_status() after running the deal (run_deal_ex()). The triggers can be overridden by calling set_trigger_override before calling run_deal_ex().

Example:

```
*     void* tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT();
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "AB05HE3");
*
*     set_engine_preference(
      PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
*     open_deal_ex(tid, pCmo);
*
*     int trg_num = get_triggers_avail(tid, NULL, NULL);
*     if(trg_num>0)
*     {
*         char *name_buf = (char*)malloc(trg_num*21);
*         char **names = (char**)malloc(trg_num * sizeof(char*));
*         for(int i = 0; i<trg_num; ++i)
*         {
*             names[i] = (char*)malloc(21);
*         }
*         signed char *status_buf = (signed char*)malloc(trg_num*MAX_PERIODS);
*         signed char **status = (signed char**)malloc(trg_num * sizeof(signed char*));
*         for(int i = 0; i<trg_num; ++i)
*         {
*             status[i] = (signed char*)malloc(MAX_PERIODS);
*         }
*         for(int i = 0; i<trg_num; i++ )
*         {
*             names[i] = &name_buf[i*21];
*             status[i] = &status_buf[i*MAX_PERIODS];
*         }
*
*         int triggers = get_triggers_avail(tid, names, NULL);
*         if(triggers <0)
*         {
*             //error handle;
*         }
*     }
*
*      close_deal_ex(tid,pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.1.4.70 void∗ CHASAPI get_user_data_for_cb (void ∗ *tid*)

Retrieves the registed user data with the WSA API set_user_data_for_cb().

Since

1.1.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Returns

A pointer to user stored data area

Example:

```
*     int fCOLLAT_ASSUMP_CB1(void* tid,
*                         char* first_period_date,
*                         int max_periods,
*                         PAY_POOL_INFO* pool_info,
*                         CCMO_COLLAT_ASSUMPTIONS* assumptions,
*                         void* user_data,
*                         char* error_message,
*                         int max_size_of_error_message
*                         );
*
*      void* tid = NULL;
*      CMO_STRUCT deal;
*      memset(&deal, 0, sizeof(CMO_STRUCT));
*      strcpy(deal.dealid,"AL2010-A");
*
*      create_deal_scenario_object(&tid,NULL,NULL,NULL);
*      open_deal_ex(tid,&deal);
*
*      set_user_data_for_cb(tid,(void*)"set_user_data_for_cb");
*      install_collat_assump_cb(tid,fCOLLAT_ASSUMP_CB1);
*      run_deal_ex(tid,&deal);
*
*      get_user_data_for_cb(tid);
*
*      close_deal_ex(tid,&deal);
*      release_deal_scenario_object(&tid);
*
```

22.1.4.71 int CHASAPI install_collat_assump_cb (void ∗ *tid*, COLLAT_ASSUMP_CB *collat_assump_cb*)

Installs collateral assumption call back function. User provided call back function will be invoked when running each collateral pool.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*collat_assump_-cb*	The user provided call back function.

Return values

0	Success
<0	Error - use get_deal_error_msg() function to obtain text of error

Example:

```
*      int collat_assump_cb_func(void* tid,
*                                char* first_period_date,
*                                int max_periods,
*                                PAY_POOL_INFO* pool_info,
*                                CCMO_COLLAT_ASSUMPTIONS* assumptions,
*                                void* user_data,
*                                char* error_message,
*                                int max_size_of_error_message)
*      {
*          if (pool_info->pool_info.loan_number == 1)
*          {
*              // check user input data
*              assert(0 == strcmp("user data what passed in", (const char *)pool_info->pool_info.usr_data))
      ;
*              std::cout << "User input data for loan 1 is:" << (const char *)pool_info->pool_info.usr_data
       << std::endl;
*
*              // do any settings you want
*              assumptions->default_type = DEFAULT_CURVE_CDR;
*          }
*          else if (pool_info->pool_info.loan_number == 2)
*          {
*              // check user input data
*              assert(123 == (int)pool_info->pool_info.usr_data);
*              std::cout << "User input data for loan 2 is:" << (int)pool_info->pool_info.usr_data <<
      std::endl;
*
*              // do any settings you want
*              assumptions->default_type = DEFAULT_CURVE_CDR;
*          }
*          else
*          {
*              // do any settings you want
*              assumptions->default_type = DEFAULT_CURVE_CDR;
*          }
*
*          return 0;
*      }
*
*      void collat_assump_cb_example()
*      {
*          void* pDeal = NULL;
*          CMO_STRUCT <em>pCmo = new CMO_STRUCT();
*          memset(pCmo, 0, sizeof(*pCmo));
*          strcpy(pCmo->dealid, "ACE06NC1");
*
*          set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*          // open deal
*          assert(0 == open_deal_ex(pDeal, pCmo));
*
*          MARKIT_POOL_INFO</em> coll_info = NULL;
*          void* coll_it = obtain_collat_iterator(pDeal, 0);
*          assert(NULL != coll_it);
*
*          // set user input data for loan 1 and loan 2
*          assert(0 == set_pool_level_user_data_for_cb(pDeal, NULL, 1, (void
      *)"user data what passed in"));   // loan 1, passed a pointer of a string
*          assert(0 == set_pool_level_user_data_for_cb(pDeal, NULL, 2, (void
      *)123));                          // loan 2, passed a int number
*
*          // get all loans
*          std::vector<MARKIT_POOL_INFO*> all_collat;
*          while((coll_info = get_next_collat(pDeal, coll_it)))
*          {
*              all_collat.push_back(coll_info);
*          }
*          assert(all_collat.size() >= 2);
*
*           // check user input data for loan 1
*           assert(1 == all_collat[0]->loan_number);
*           assert(0 == strcmp("user data what passed in", (const char *)all_collat[0]->usr_data));
*
*           // check user input data for loan 2
*           assert(2 == all_collat[1]->loan_number);
*           assert(123 == (int)all_collat[1]->usr_data);
*
*           // install call back function
*           assert(0 == install_collat_assump_cb(pDeal, collat_assump_cb_func));
*
*           // run deal, call back func will be triggered many times for each loan
*           assert(0 == run_deal_ex(pDeal, pCmo));
*
*           assert(0 == close_deal_ex(pDeal, pCmo));
*           delete pCmo;
*           pCmo = NULL;
*       }
*
```

See Also

install_per_period_assump_cb()

22.1.4.72 void CHASAPI install_input_dir_callback (INPUT_DIR_CB *callback*)

Installs input_path call back function. User provided call back function will be invoked to set extra input directory before SFW or CHS deal archive files (.SFW, .CHS, .LLD, etc) are accessed

Since

1.6.0

Availability CHS, SFW

Parameters

in	*callback*	The user provided call back function. The call back function type should be be INPUT_DIR_CB. To uninstall the call back function, call with callback set to NULL.

Returns

None

Note

Calling to this api is not thread-safe.

Example:

```
*       // The CB function
*       USER_CB_RETURN_CODE DealDirCB(
*                               const char* deal_name,
*                               const char* file_name,
*                               const char* archive_name,
*                               const char* default_dir_if_noop,
*                               char* file_dir_to_provide,
*                               int size_of_file_dir_buffer,
*                               char* error_msg,
*                               int size_of_error_msg_buffer
*                               );
*       // should be called in main thread before other threads start
*       install_input_dir_callback(DealDirCB);
*
```

22.1.4.73 void CHASAPI install_input_dir_callback_ex (INPUT_DIR_CB_EX *callback_ex*)

Installs input_path call back function Ex. User provided call back function will be invoked to set extra input directory before SFW or CHS deal archive files (.SFW, .CHS, .LLD, etc) are accessed

Since

1.6.0

Availability CHS, SFW

Parameters

in	*callback*	The user provided call back function. The call back function type should be be INPUT_DIR_CB. To uninstall the call back function, call with callback set to NULL.

Returns

None

Note

Calling to this api is not thread-safe.

Example:

```
    *       // The CB function
    *       USER_CB_RETURN_CODE DealDirCBEx(
    *                               const char* deal_name,
    *                               const char* file_name,
    *                               const char* archive_name,
    *                               const char* default_dir_if_noop,
    *                               char* file_dir_to_provide,
    *                                const char* update_date,
    *                               int size_of_file_dir_buffer,
    *                               char* error_msg,
    *                               int size_of_error_msg_buffer
    *                               );
    *       // should be called in main thread before other threads start
    *       install_input_dir_callback_ex(DealDirCBEx);
    *
```

22.1.4.74 int CHASAPI install_per_period_assump_cb (void ∗ *tid,* PER_PERIOD_ASSUMP_CB *per_period_assump_cb*)

Installs per period assumption call back function. User provided call back function will be invoked when running each period.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*per_period_-assump_cb*	The user provided call back function. The call back function type should be be PER_PERIOD_ASSUMP_CB.

Return values

0	Success
<0	Error - use get_deal_error_msg() function to obtain text of error

Example:

```
*   int fPER_PERIOD_ASSUMP_CB(  void* tid,
*                               int period,
*                               int max_periods,
*                               PAY_POOL_INFO* pool_info,
*                               PAY_POOL_STATE* pool_state,
*                               CCMO_PERIOD_ASSUMPTIONS* assumptions,
*                               void* user_data,
*                               char* error_message,
*                               int max_size_of_error_message
*                            );
*
*   void *tid = NULL;
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   strcpy(pCmo->dealid,"AL2010-A");
*
*   int iret = open_deal_ex(tid, pCmo);
*
*   iret = install_per_period_assump_cb(tid, fCOLLAT_ASSUMP_CB);
*
*   iret = run_deal_ex(tid, pCmo);
*
*   iret = close_deal_ex(tid, pCmo);
*   delete pCmo;
*   pCmo = NULL;
*
```

See Also

install_collat_assump_cb()

22.1.4.75 int CHASAPI install_pool_cashflow_cb (void ∗ *tid*, POOL_CASHFLOW_CB *pool_cashflow_cb*)

This function installs the user defined pool cashflow callback function.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*pool_cashflow_-cb*	The user callback function for pool cash flow

Return values

0	Success

<0	Error - for details call get_deal_error_msg()

Example:

```
*
*     void fPOOL_CASHFLOW_CB(void* tid,
*                           PAY_POOL_INFO* pool_info,
*                           int last_populated_period,
*                           PAY_POOL_STATE* pool_cashflow,
*                           CCMO_PERIOD_ASSUMPTIONS* assumptions,
*                           void* user_data
*                           );
*
*      void *tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(CMO_STRUCT));
*      char deal_id[] = "AB05HE3";
*      int len = sizeof(deal_id) / sizeof(char);
*      memcpy(pCmo->dealid,deal_id,len);
*
*      open_deal_ex(tid, pCmo));
*
*      set_user_data_for_cb(tid,(void*)"user data for pool cashflow cb");
*
*      install_pool_cashflow_cb(tid, fPOOL_CASHFLOW_CB);
*
*      run_deal_ex(tid,pCmo);
*      close_deal_ex(tid,pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.1.4.76 int CHASAPI install_user_cleanup_cb (void ∗ *tid,* USER_CLEANUP_CB *user_cleanup_cb,* int *invoke_on_deal_close*)

Installs user clean up function

Since

1.1.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*user_cleanup_-cb*	User provided call back function
in	*invoke_on_deal-_close*	Indicate if call back function will be invoked. • 1: The call back function will be invoked. • 0: The call back function won't be invoked.

Return values

0	Success
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Note

If user registers some user data with the WSA API, she might want the WSA API to provide a way to release the data at appropriate time. The WSA API allows a user to register clean up call back function which will be called when deal is closed or when new user data is provided. The pointer to the previously registered user data will be passed to this function allowing the user to perform a proper release of resources associated with user data. Calling install_user_cleanup_cb() is a way to register this call back The signature of the call back function see USER_CLEANUP_CB.

Example:

```
*      int fCOLLAT_ASSUMP_CB1(void* tid,
*                             char* first_period_date,
*                             int max_periods,
*                             PAY_POOL_INFO* pool_info,
*                             CCMO_COLLAT_ASSUMPTIONS* assumptions,
*                             void* user_data,
*                             char* error_message,
*                             int max_size_of_error_message
*                             );
*
*       void fUSER_CLEANUP_CB(void* user_data);
*
*       void* tid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"AL2010-A");
*
*       create_deal_scenario_object(&tid,NULL,NULL,NULL);
*       open_deal_ex(tid,&deal);
*
*       set_user_data_for_cb(tid,(void*)"set_user_data_for_cb");
*       install_collat_assump_cb(tid,fCOLLAT_ASSUMP_CB1);
*
*       install_user_cleanup_cb(tid,fUSER_CLEANUP_CB,0);
*
*       run_deal_ex(tid,&deal);
*       close_deal_ex(tid,&deal);
*       release_deal_scenario_object(&tid);
*
```

22.1.4.77 long CHASAPI is_credit_sensitive_ex (void * *tid,* long *loan_num*)

Indicates whether defaults and delinquencies can affect the currently open deal.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_num*	Reserved for future use, must be -1

Return values

1	Is credit sensitive (affected be defaults and/or delinquencies)
0	Is not credit sensitive
-1	Deal is not open
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       // Deal is already opened
*
*       // If iR is 1, the deal is credit sensitive.
*       int iR = is_credit_sensitive_ex(tid, -1);
*
```

22.1.4.78 void∗ CHASAPI obtain_collat_iterator (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*)

This function obtains a pointer to the internal to the WSA API collateral iterator. This pointer should be passed to consecutive calls to get_next_collat() to retrieve collateral information.

Keep in mind that the second call to this function for the same deal will invalidate all the collateral pointers retrieved from the WSA API during the first call.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier.
in	*reremic_deal_id-_or_null*	Pass 0 for main deal or remic name for collateral of the child deal

Return values

Pointer	to be passed to get_next_collat() function
0	Error

Example:

```
*       void * tid = NULL;
*       // deal has been opened
*
*       MARKIT_POOL_INFO* coll_info =0;
*       void* coll_it = obtain_collat_iterator(tid, 0);
*       if(coll_it == 0)
*       {
*           std::cout << "Failure to start collat iteration " <<
      get_deal_error_msg(tid) << std::endl;
*       }
*       while(coll_info = get_next_collat(tid,coll_it))
*       {
*           // do what you need with collateral
*       }
*
```

22.1.4.79 long CHASAPI open_deal_ex (void ∗ *tid*, CMO_STRUCT ∗ *cmos*)

Opens a deal so that it can be run. This method must be called before calling any functions that change the setting of a specified deal, run or analyze a specified deal.

Since

0.9.0

Availability ALL

Precondition

set_input_path() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in,out	*cmos*	A pointer to a structure used to pass information to the API and return descriptive information regarding the deal. e.g. next_pay_date, first_pay_date, orig_settlement_date, periodicity, num_bond, num_colls.

Return values

0	Deal opened successfully
<0	Error - Deal opened failed, check details by calling get_deal_error_msg()

Example:

```
*       void* tid(NULL);
*        //Declare the cmos structure for the deal and initialize it.
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "BAA05006";     // also can be a CUSIP/ISIN of SFW/CDOnet bond, e.g. "31364JMQ3",
        "000759BP4", "00969QAJ0", "US000780BW56", "XS0333236890"
*       int len = sizeof(pCmo->dealid) / sizeof(pCmo->dealid[0]);
*       strncpy(pCmo->dealid,deal_id,len-1);
*
*       set_input_path("C:/deals");
*       open_deal_ex(tid, pCmo);
*
*       run_deal_ex(tid, pCmo);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

Structure cmos should be allocated and initialized by the user. The following fields in cmos should be set before opening the deal:

- dealid (Required, this field also can be a CUSIP or ISIN. If the deal id is unknown, it can be found by calling get_moodys_id() with CUSIP or ISIN.)
- bondid (Optional. If not set the first bond in the deal will be used.)
- bond.cusip (Optional. This is optional for opening deals but required for opening agency pools from LPD files.) cusip is a 9-character alphanumeric code which uniquely identifies a security. bond.cusip is a char array. If it is not provided, user should leave it empty(bond.cusip[0] == 0). e.g."00252FAD3" is a legal cusip.
- actual_col (Required. 0 or 1. If 1, actual collateral is used. If 0 collateral is bucketed, which greatly speeds up processing while maintaining a high degree of accuracy.)
- settlement_date (Optional. This is the deal status date from which projections will be run. If not set, the most recent deal status will be used. mm/dd/yy.)
- first_projected_date (Optional. If set to ("1"), the deal opened based on as of date and the bond's period-0 payment date)
- bond.id (Optional. If specify with "EXACT", bondid is required a valid value, otherwise, open deal would fail.)

$**$ If not running custom amortization and not setting scenarios, call set_custom_amortization_ex(tid,CUSTOM_AMORT_NONE) after open_deal_ex() to ensure the deal is properly run.

22.1.4.80 long CHASAPI open_pool_from_file (void ∗ *tid,* const char ∗ *cusip,* const char ∗ *reserved,* int *YYYYMMDD_settlement_date*)

This function opens a single pool.

Since

1.1.0

Availability ALL

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*cusip*	The CUSIP being requested.
in	*reserved*	Use NULL.
in	*YYYYMMDD_-settlement_date*	Settlement date in YYYYMMDD format.

Return values

0	SUCCESS
-99	Error, for details call get_deal_error_msg()

Example:

```
*     void* tid = NULL;
*     int iret = create_deal_scenario_object(&tid,NULL,NULL,NULL);
*     set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*
*     iret = open_pool_from_file(tid, "03072SB38", 0, 20120901);
*     if(iret <0)
*     {
*       //error handle;
*     }
*
```

22.1.4.81 int CHASAPI price_bond (void ∗ *tid,* const char ∗ *bondid,* PRICING_ANCHORS *anchorType,* double *anchorValue,* PRICING_RESULTS ∗ *results*)

This function calculates cashflow analytics for a given bond. It should be called after running cashflow

Since

1.2.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The bond identifier.
in	*anchorType*	Type of anchor for pricing, available options: PRICE, YIELD, or DM
in	*anchorValue*	Value of the provided pricing anchor
out	*results*	Calculated bond analytics

Return values

=0	Successful
<0	Failed, check details by calling get_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       open_deal_ex(tid, pCmo);
*
*       double dAnchor             = 2.0000;
*       PRICING_ANCHORS anchorType = YIELD;
*       char *szBondid             = "A1";
*       PRICING_RESULTS results;
*       memset(&results, 0x00, sizeof(PRICING_RESULTS));
*       //validate, before run_deal_ex(), call price_bond
*       assert(-99 == price_bond(tid, szBondid, anchorType, dAnchor, &results));
*
*       assert(0 == run_deal_ex(tid,pCmo));
*
*       memset(&results, 0x00, sizeof(PRICING_RESULTS));
*       int nRet = price_bond(tid, szBondid, anchorType, dAnchor, &results);
*       if(nRet !=0)
*       {
*           //error handle;
*       }
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.82 int CHASAPI release_deal_scenario_object (void ** *Tid*)

This function releases a deal_scenario object (dso) and frees the resources. Once the dso is released, no further processing can be performed with it.

Since

0.9.0

Availability ALL

Precondition

create_deal_scenario_object() has been called.

Parameters

in	*Tid*	The deal/scenario identifier obtained when the dso was created.

Return values

0	Success
-1	Error - Invalid Parameter
-2	Error - Runtime error, memory error or other error

Example:

```
*    void* ptid = NULL;
*    CMO_STRUCT *pCmo = new CMO_STRUCT();
*    strcpy(pCmo->dealid, "AL2010-A");
```

```
*
*   bool Debug = true;
*   short LogAction = Debug ? LOG_OPTION_OVERWRITE :
      LOG_OPTION_SUPPRESS;
*   int iret = creat_deal_scenario_object(&ptid, &LogAction, "log.txt", &Debug);
*   if(iret != 0)
*   {
*       return;
*   }
*   open_deal_ex(ptid, pCmo);
*
*   release_deal_scenario_object(&ptid);
*
*   close_deal_ex(ptid, pCmo);
*   delete pCmo;
*   pCmo = NULL;
*
```

See Also

create_deal_scenario_object()

22.1.4.83 int CHASAPI replace_collateral (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, MARKIT_POOL_INFO ∗ *collat_array[ ]*, int *collat_array_size*)

This function replaces the collateral allocated by the WSA API during loading stage of the deal with collateral prepared by the user. It allows the user to perform customized aggregation when use in combination with collateral iterator.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario identifier obtained when the dso was created.
in	*reremic_deal_id-_or_null*	0 for parent deal or name of the child deal
in	*collat_array*	User allocated array of collateral information
in	*collat_array_size*	Number of collaterals in the array

Return values

0	No error
<0	Error: Call the get_deal_error_msg() function for more details

Note

This function might be called after deal is loaded to replace collateral read from the files with the one provided by user. User has to provide collateral for every group in the deal. User is responsible for allocating and releasing the memory passed to the collat_array parameter. The WSA API will allocate its own memory and copy information provided by the user.

Example:

```
*      void *tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(CMO_STRUCT));
*      char deal_id[] = "AL2010-A";
*      int len = sizeof(deal_id) / sizeof(char);
*      memcpy(pCmo->dealid,deal_id,len);
*
*      replace_collateral(tid, NULL, NULL, 0);
*
```

```
*       open_deal_ex(tid,pCmo);
*
*       std::vector<MARKIT_POOL_INFO*> collat_vector;
*       MARKIT_POOL_INFO* coll_info =0;
*       void* coll_it = obtain_collat_iterator(tid, 0);
*
*       while(coll_info = get_next_collat(tid,coll_it))
*       {
*           if(!coll_info->arm && coll_info->type != REMIC)
*           {
*               int agency_type = coll_info->type;
*               double wacAddition = 0;
*               switch(agency_type)
*               {
*               case GNMA1:
*                   wacAddition = 0.5;
*                   break;
*               case GNMA2:
*                   wacAddition = 1.5;
*                   break;
*               default:
*                   wacAddition = 2.5;
*                   break;
*               }
*               wacAddition /= 100;
*               double wac = coll_info->psa_coupon + wacAddition;
*               coll_info->wac = wac;
*               coll_info->gross_coupon = wac;
*               int wala = coll_info->wala;
*               coll_info->wala =0;
*               if(coll_info->prin_lockout > 0)
*                   coll_info->prin_lockout += wala;
*           }
*           collat_vector.push_back(coll_info);
*       }
*
*       replace_collateral(tid, NULL, &collat_vector[0], -1);
*       collat_vector.clear();
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.84 int CHASAPI replace_collateral_for_managed_code (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* MARKIT_POOL_INFO *collat_array[ ],* int *collat_array_size*)

This function replaces the collateral allocated by the WSA API during loading stage of the deal with collateral prepared by the user. It allows the user to perform customized aggregation when use in combination with collateral iterator.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario identifier obtained when the dso was created.
in	*reremic_deal_id-_or_null*	0 for parent deal or name of the child deal
in	*collat_array*	User allocated array of collateral information
in	*collat_array_size*	Number of collaterals in the array

Return values

0	No error
<0	Error: Call the get_deal_error_msg() function for more details

Note

This function might be called after deal is loaded to replace collateral read from the files with the one provided by user. User has to provide collateral for every group in the deal. User is responsible for allocating and releasing the memory passed to the collat_array parameter. The WSA API will allocate its own memory and copy information provided by the user.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "AL2010-A";
*       int len = sizeof(deal_id) / sizeof(char);
*       memcpy(pCmo->dealid,deal_id,len);
*
*       open_deal_ex(tid,pCmo);
*
*       std::vector<MARKIT_POOL_INFO> collat_vector;
*       MARKIT_POOL_INFO* coll_info =0;
*       void* coll_it = obtain_collat_iterator(tid, 0);
*
*       while(coll_info = get_next_collat(tid,coll_it))
*       {
*           if(!coll_info->arm && coll_info->type != REMIC)
*           {
*               int agency_type = coll_info->type;
*               double wacAddition = 0;
*               switch(agency_type)
*               {
*               case GNMA1:
*                   wacAddition = 0.5;
*                   break;
*               case GNMA2:
*                   wacAddition = 1.5;
*                   break;
*               default:
*                   wacAddition = 2.5;
*                   break;
*               }
*               wacAddition /= 100;
*               double wac = coll_info->psa_coupon + wacAddition;
*               coll_info->wac = wac;
*               coll_info->gross_coupon = wac;
*               int wala = coll_info->wala;
*               coll_info->wala =0;
*               if(coll_info->prin_lockout > 0)
*                   coll_info->prin_lockout += wala;
*           }
*           collat_vector.push_back(*coll_info);
*       }
*
*       replace_collateral_for_managed_code(tid, NULL, &collat_vector[0],
        -1);
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.85 long CHASAPI run_deal_ex (void * *tid,* CMO_STRUCT * *cmos*)

Projects cashflows for the currently open deal using specified scenario. The scenarios (such as market rates, prepayment assumptions, default rates, etc.) can be set using the corresponding functions before running the deal.

Since

0.9.0

Availability ALL

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in,out	*cmos*	A pointer to the CMO_STRUCT used in open_deal_ex().

Return values

$>=$	0 Deal run successfully
-1	Error - Deal not open
-2	Error - Unable to run deal
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Note

Scenario information (market rates, prepayment assumptions, credit sensitivity assumptions, etc) should be set after opening the deal and before calling run_deal_ex(). The same deal can be re-run multiple times, varying the scenarios, without re-opening the deal. It is fastest to run one deal at a time, varying the scenarios, than to run one scenario at a time, varying the deal.

Example:

```
*       void* tid(NULL);
*       CMO_STRUCT *pCmo = new CMP_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "BAA05006";
*       int len = sizeof(pCmo->dealid) / sizeof(pCmo->dealid[0]);
*       strncpy(pCmo->dealid,deal_id,len-1);
*
*       set_input_path("C:/deals");
*       open_deal_ex(tid, pCmo);
*
*       run_deal_ex(tid, pCmO);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.86 long CHASAPI set_addit_group_delinquencies (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int *group_number,* short *is_vector,* int *delinq_type,* double ∗ *dqVal*)

Sets the group level delinquency for a specific type.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal ID if child deal. Otherwise pass null.

in	*group_number*	The group ID to apply defaults to, -1 means apply delinquency assumption on deal level.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*delinq_type*	Delinquent type, must be one of GROUP_DELINQ_STATES. SFW and CD-Onet can only support GROUP_DELINQ_90 now.
in	*dqVal*	Delinquent data.

Return values

1	SFW or CDOnet Engine Warning: Only GROUP_DELINQ_90 rates are used by the engine currently.
0	Assumption set successfully.
-1	Error - Deal not opened
-2	Invalid parameter.
-99	Fail - check details by calling get_deal_error_msg()

Example:

```
*   void* tid = NULL;
*   CMO_STRUCT *pCmo = new CMO_STRUCT();
*   memset(pCmo, 0, sizeof(CMO_STRUCT));
*   char deal_id[] = "WFNMT";
*   int len = sizeof(deal_id) / sizeof(char);
*   memcpy(pCmo->dealid, deal_id, len);
*
*   set_input_path("the_input_path");
*   assert(0 == open_deal_ex(tid, pCmo));
*
*   double delinq[3] = {0.5, 0.3, 0.2};
*   assert(0 == set_addit_group_delinquencies(tid, NULL, -1, sizeof(delinq)/
      sizeof(delinq[0]), GROUP_DELINQ_90, delinq));
*
*   assert(0 == run_deal_ex(tid, pCmo));
*
*   close_deal_ex(tid, pCmo);
*   delete pCmo;
*   pCmo = NULL;
*
```

22.1.4.87 int CHASAPI set_bond_cf_mode (void ∗ *tid*, BOND_CF_MODE *mode*, BOND_PAYMENT_DATES_TYPE *payment_dates_type*, int *propagate_to_remics*)

Sets the bond cash flow mode and payment dates type.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*mode*	Cashflow generating mode. Must be one of BOND_CF_MODE.
in	*payment_dates-_type*	Bond payment date type. Must be one of BOND_PAYMENT_DATES_TYPE.
in	*propagate_to_-remics*	A flag to indicate if set the same for the underlying remic deals.

Return values

0	No error
<0	Error - check details by calling get_deal_error_msg()

Warning

- If this api is not called
  - CHS engine will use BOND_CF_MODE_TRADING as bond cf mode
  - SFW engine will use BOND_CF_MODE_HOLDING as bond cf mode
  - CDOnet engine will use BOND_CF_MODE_TRADING as bond cf mode
- For payment_dates_type
  - It applies to CHS deal only
  - SFW and CDOnet deal will use its internal logic to determine payment dates for a bond.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AL2010-A");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       open_deal_ex(tid, pCmo);
*
*       int ret =  set_bond_cf_mode(tid,BOND_CF_MODE_TRADING,
      BOND_PAYMENT_DATES_THEORETICAL,0);
*       if(ret!=0)
*       {
*           //error handle;
*       }
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.88 short CHASAPI set_bond_flow (void * *tid,* const char * *bondid,* int *flow_identifier,* short *flow_length,* double * *flows*)

Sets the specified bond cashflow to the values passed (in a vector) for the (see price_bond()) the number of periods specified (all later periods will be set to zero (0).)

Since

2.1.0

Availability CDOnet, CHS, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The bond identifier.
in	*flow_identifier*	Identifies the requested cash flow. • For CDOnet and SFW, must be one of the following bond cashflow identifiers: • FLOW_BOND_BALANCE • FLOW_BOND_INTEREST • FLOW_BOND_PRINCIPAL • FLOW_BOND_CAPPED_RATE • FLOW_BOND_RATE • For CHS must be one of the following bond cashflow identifiers: • FLOW_BOND_BALANCE • FLOW_BOND_INTEREST • FLOW_BOND_PRINCIPAL • FLOW_BOND_FLT_INDEX • FLOW_BOND_PRINCIPAL_WRITEDOWN • FLOW_BOND_INTEREST_SHORTFALL
in	*flow_length*	The number of values being passed in the array "flows".
in	*flows*	A pointer to an array of cash flows.

Return values

1	No error
-1	Deal not open
-2	Invalid bond id
-3	Invalid cashflow type
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void * pDeal = NULL;
*       //Deal have opened
*
*       double balance[] = {12020209.49,11997482.94,11974643.97,11951692.01,11928626.51,0.0};
*       double principal[] = {0,22726.54,22838.90,22951.95,23065.50,0.0};
*       double interest[] = {0,5092.46,2899.39,2893.872,2888.32,0.0};
*       assert(0 == set_bond_flow(pDeal, "A1", FLOW_BOND_BALANCE, sizeof(
      balance)/sizeof(double), balance));
*       assert(0 == set_bond_flow(pDeal, "A1", FLOW_BOND_INTEREST, sizeof(
      interest)/sizeof(double), interest));
*       assert(0 == set_bond_flow(pDeal, "A1", FLOW_BOND_PRINCIPAL, sizeof(
      principal)/sizeof(double), principal));
*
```

Warning

If user want to get the price results with the custom bond cashflow, this function should be called befor price-_bond()

22.1.4.89 int CHASAPI set_cleanup_call (void ∗ *tid*, double ∗ *percentage*, CLEAN_UP_CALL_BALANCE_TYPE ∗ *call_balance_type*, CLEAN_UP_CALL_LINK_TYPE ∗ *link_type*, int ∗ *yyyymm_date*, BOOLYAN *set_sup_remic*)

This method is to set options of cleanup call.

Since

3.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*percentage*	The percentage of balance to call. It must be a decimal between 0 and 1.
in	*call_balance_-type*	Type of call balance. Should be one of CLEAN_UP_CALL_BALANCE_TYPE.
in	*link_type*	Type of call link. Should be one of CLEAN_UP_CALL_LINK_TYPE.
in	*yyyymm_date*	The specified date to call(of format YYYYMM).
in	*set_sup_remic*	Settings are applied to underlying deals if TRUE. Otherwise, it will not.

Return values

0	Success.
-2	Error - Invalid parameter.
-99	Error ,call get_deal_error_msg() for details.

Example:

```
*     void * pDeal = NULL;
*     // Open deal
*
*     assert(0 == set_call_option(pDeal, CLEANUP_CALL, false));
*     char call_date[11]     =  {0};
*     double call_percent    =  0.0;
*     int call_percent_calc =  0;
*     int iRet = get_cleanup_call_ex(pDeal, call_date, &call_percent, &
     call_percent_calc);
*
*     double pct = 0.8;
*     int callDate = 201512;
*     CLEAN_UP_CALL_LINK_TYPE linkType =
     CLEAN_UP_CALL_LINK_AND;
*     assert(0 == set_cleanup_call(pDeal, &pct, NULL, &linkType, &callDate, false));
*
*     iRet = get_cleanup_call_ex(pDeal, call_date, &call_percent, &call_percent_calc);
*
```

Note

For CHS engine, the call balance type only supports the "END_COLLAT_BAL".

22.1.4.90 short CHASAPI set_collateral_flow_ex (void ∗ *tid*, long *group_number*, int *flow_identifier*, short *flow_length*, double ∗ *flows*, CMO_STRUCT ∗ *cmo*)

Sets the specified collateral cashflow to the values passed (in a vector) by the calling program for the number of periods specified (all later periods will be set to zero (0).)

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*group_number*	The collateral group for which cashflows are requested, 0 for total (deal level).
in	*flow_identifier*	Identifies the requested cash flow. Must be one of the collateral cashflow identifiers (see EXTENDED_FLOW_COLLATERAL_IDENTIFIER).
in	*flow_length*	The number of values being passed in the array "flows".
in	*flows*	A pointer to an array of cash flows.
in	*cmo*	The pointer to the CMO_STRUCT used when opening the deal.

Return values

1	No error
-1	Deal not open
-2	Invalid cashflow type
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Note

set_custom_amortization_ex() must be called before this function.

Example:

```
*      void *tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(CMO_STRUCT));
*      char deal_id[] = "AB05HE3";
*      int len = sizeof(deal_id) / sizeof(char);
*      memcpy(pCmo->dealid, deal_id, len);
*
*      open_deal_ex(tid,pCmo);
*      run_deal_ex(tid,pCmo);
*
*      set_custom_amortization_ex(tid,
      CUSTOM_AMORT_BY_GROUP);
*
*      set_collateral_flow_ex(tid, 0, 100, 0, NULL, pCmo);
*
*      close_deal_ex(tid,pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.1.4.91 short CHASAPI set_custom_amortization_ex (void ∗ *tid*, short *newVal*)

Sets the deal to use custom collateral amortization and performs various initializations. This must be called before calling set_collateral_flow_ex().

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*newVal*	The type of custom amortization. Must be one of the following: • CUSTOM_AMORT_NONE: WSA API amortizes the collateral. • CUSTOM_AMORT_BY_DEAL: Cashflows are set at the deal level (collateral group 0),SFW engine not supported deal level currently. • CUSTOM_AMORT_BY_GROUP: Cashflows are set by collateral group.

Return values

0	No error
-1	Deal not open
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Note

This must be called before set_collateral_flow_ex. SFW engine not supported deal level(CUSTOM_AMORT_BY_DEAL)currently.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "AL2010-A";
*       int len = sizeof(deal_id) / sizeof(char);
*       memcpy(pCmo->dealid,deal_id,len);
*
*       open_deal_ex(tid,pCmo);
*       run_deal_ex(tid,pCmo);
*
*       set_custom_amortization_ex(tid,
       CUSTOM_AMORT_BY_GROUP);
*
*       set_collateral_flow_ex(tid, 0, 100, 0, NULL, pCmo);
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.92 void CHASAPI set_deal_calc_level (void ∗ *tid,* CALC_LEVEL *level,* int *propagate_to_remics*)

Sets the deal calculation level: CALC_LEVEL_BASIC or CALC_LEVEL_FULL.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*level*	CALC_LEVEL_BASIC or CALC_LEVEL_FULL,some bond/collateral flow only be calculated when set to CALC_LEVEL_FULL.

in	*propagate_to_-remics*	If apply to remics as well

Returns

None

Example:

```
*       void * pDeal = NULL;
*       // Open deal
*
*       set_deal_calc_level(pDeal,CALC_LEVEL_BASIC,0);
*
```

See Also

get_bond_flow_ex get_bond_flow_ex1 get_bond_flow_ex1_for_managed_code get_collateral_flow_ex get_-collateral_flow_ex1

22.1.4.93 void CHASAPI set_deal_error_msg (void ∗ *tid,* const char ∗ *err*)

Sets the error message for the deal.

Since

0.9.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*err*	The error message.

Return values

0	No error
<0	Fail - check details by calling get_deal_error_msg()

Example:

```
*       void *tid = NULL;
*       char err[100] = {0};
*       set_deal_error_msg(tid, err);
*
```

22.1.4.94 void CHASAPI set_deal_search_mode (DEAL_SEARCH_MODE *mode*)

Sets the mode for deal search: either DEAL_SEARCH_FROM_FILE or DEAL_SEARCH_FROM_MEMORY.

Availability ALL

Parameters

in	*mode*	DEAL_SEARCH_FROM_FILE or DEAL_SEARCH_FROM_MEMORY

Warning

This method is obsolete and does *NOT* affect deal search.

Deprecated This method is deprecated and does *NOT* affect deal search.

22.1.4.95 void CHASAPI set_default_from_ex (void ∗ *tid,* const short *type,* BOOLYAN *set_sup_remic*)

Sets which balance defaults are calculated from. The setting will also apply to underlying deals if set_sup_remic is TRUE. The default value is DEFLT_FROM_CURBAL.

Since

1.1.0

Availability SFW, CDOnet, CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	The type of balance the defaults are calculated from. • DEFLT_FROM_CURBAL: The current balance for the period. SFW and CDOnet engine can only support this type. • DEFLT_FROM_ORIGBAL: The balance in period 0 (deal "as of" date). • DEFLT_FROM_ZERO_CPR: The scheduled balance if no prepayments and no defaults.
in	*set_sup_remic*	Settings are applied to underlying deals if TRUE.

Returns

None

Example:

```
*       // Deal is already opened
*
*       // Set the defaults from the original balance. If the defaults
*       // are a constant, the dollar amount will default each period.
*       // Apply this to all underlying deals
*       set_default_from_ex(tid, DEFLT_FROM_ORIGBAL, TRUE);
*
```

Warning

when it is SFW or CDOnet deal, type could only support DEFLT_FROM_CURBAL.

22.1.4.96 long CHASAPI set_defaults_ex (void ∗ *tid,* short *type,* short *is_vector,* double ∗ *pval,* long *loan_num,* BOOLYAN *set_sup_remic*)

Sets the constant or vectored default rate and type that will be used for the pool specified by loan_num, with the ability to apply to underlying deals if it is a reremic.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	Type of default curve. Must be one of: • DEFAULT_CURVE_CDR - Constant Default Rate(CDR): The Constant Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the year assumed to terminate during the year through events of default. It is mathematically related to MDR: (1 – MDR)^12 = 1 – CDR. Seasoning of mortgages/loans will not be accounted for if a vectored default rate is set. • DEFAULT_CURVE_SDA - Standard default curve: Measuring for defaults in the residential mortgage market The SDA Standard Default Assumptions rate specifies an annual default percentage as a function of the seasoning of the mortgages/loans. 100% SDA assumes default rates of 0.02% CDR in the first month following origination of the mortgage loans and an additional 0.02% CDR in each succeeding month until the 30th month. In the 30th month and beyond, 100% SDA assumes a fixed annual default rate of 0.6% CDR. Any SDA other than 100% is calculated by multiplying the annual default rate (the CDR) by that multiple of 100%. • DEFAULT_CURVE_MDR - Monthly Default Rate The Monthly Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the month assumed to terminate during the month through events of default. It is mathematically related to CDR: (1 – MDR)^12 = 1 – CDR. Seasoning of mortgages/loans will not be accounted for if a vectored default rate is set. The engine will start with the first entry of the default vector in forecasting default. • DEFAULT_CURVE_SEASONED_CDR - The Constant Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the year assumed to terminate during the year through events of default. It is mathematically related to MDR: (1 – MDR)^12 = 1 – CDR. Seasoning of mortgages/loans will be accounted for if a vectored default rate is set. • DEFAULT_CURVE_SEASONED_MDR - The Monthly Default Rate is the percentage of the mortgages/loans outstanding at the beginning of the month assumed to terminate during the month through events of default. It is mathematically related to CDR: (1 – MDR)^12 = 1 – CDR. Seasoning of mortgages/loans will be accounted for if a vectored default rate is set. The engine will set the projection starting point in the default vector by the age of the asset. • DEFAULT_CURVE_PCT - The PCT is similar to the MDR curve except that defaults are applied each month to the period 0 balance of the loan, rather than the current balance of the loan. This option is only available in SFW and CDOnet engine. In CHS engine, this option is equivalent to use option DEFAULT_CURVE_MDR with setting DEFLT_FROM_ORIGBAL in function set_default_from_ex(). • DEFAULT_CURVE_PLD - Prodject Loan Default Rate: The PLD rates specifies the annum default percentage of the then-outstanding principal balance of each of the Mortgage Loans in relation to its loan age. 100% PLD represents 100% of such assumed rate of involuntary prepayments. This option is only available in SFW and CHS engine.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.

in	*pval*	A pointer to the default rates (or rate). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will be applied to underlying deals. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened
*
*       double rate;    // default speed
*       short type;     // default type
*
*       // Set 8% CDR for whole deal
*       // And apply to underlying
*       type = DEFAULT_CURVE_CDR;
*       rate = .08;
*       set_defaults_ex(ptid, type, 0, &rate, -1, true);
*
*       // Set vector of MDR for second piece of collateral.
*       // Do NOT apply to underlying. rates is a previously.
*       double rate_array[]={.01,.02,.03,.04,.05};
*       set_defaults_ex(ptid, DEFAULT_CURVE_MDR, 5, rate_array, 2, false);
*
*       run_deal_ex(ptid,&deal);
*       close_deal_ex(ptid,&deal);
*
```

Note

- Default rates are expressed as decimals. 5.25% would be .0525.
- If loan_num = -1, the setting will apply to all collateral loans. If loan_num is 1-based index, the setting will only apply to the loan specified by loan_num.
- value for current period (0-indexed element) will not be applied.

See Also

- set_defaults_ex can be used to set default rates for both top level deals and underlying deals
- set_reremic_defaults is used to set default rates for underling deal. If the deal specified by para dealid is not an underlying deal, it will return an error.

22.1.4.97 void CHASAPI set_error_handling_level (ERROR_HANDLING_LEVEL *level*)

Sets the error handling level for processing: log error message or stop running.

Since

0.9.0

Availability ALL

Parameters

`in`	*level*	ERROR_HANDLING_LEVEL_LOG_IT or ERROR_HANDLING_LEVEL_ST-OP_CALCULATION • ERROR_HANDLING_LEVEL_LOG_IT: Log error message to log file when error happens. • ERROR_HANDLING_LEVEL_STOP_CALCULATION : Stop running when error happens.

Returns

None

Example:

```
*   set_error_handling_level(ERROR_HANDLING_LEVEL_LOG_IT
      );
*
*   void* ptid=NULL;
*   // open deal
*
```

22.1.4.98 void CHASAPI set_input_path (const char ∗ *input_path*)

Sets the path of the deals' data files. The method only sets the path to the input parameter but does NOT check the validity of the input.

Since

0.9.0

Availability ALL

Parameters

`in`	*input_path*	This is the fully qualified path to the data files. The ending forward slash or backslash (depending on the operating system) is optional. Linux format:" /temp/deals". Windows format:"c:\\tmp\\deals".

Returns

None

Example:

```
*       // Must be called before opening a deal.
*       // Data located in C:\data\chs
*       set_input_path("C:\\data\\chs");
*
*       void * ptid = NULL;
*       // Declare the main structure for the deal and initialize it.
*       CMO_STRUCT * pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       // Deal is 02-010E (file is 02-001E.chs)
*       strcpy(pCmo->dealid, "02-010E");
*       // Use bucketed collateral
*       pCmo->actual_coll = 0;
*       // Set the "as of" date for Sept 9, 2009 (will open the most recent)
*       strcpy(pCmo->settlement_date, "09/09/09");
*       // Set the initial bond to blank. The first bond will be used.
*       strcpy(pCmo->bondid, "");
*       // Open the deal
*       if(0 > open_deal_ex(ptid, pCmo))
*       {
```

```
*          delete pCmo;
*          return -1;
*      }
*
*      close_deal_ex(ptid, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

This function only stores the input_path string. If the path is invalid, open_deal_ex() will return error.

22.1.4.99 int CHASAPI set_log_options (void ∗ *tid,* short ∗ *LogAction,* char ∗ *LogFile,* BOOLYAN ∗ *Debug*)

Sets the log/message options either globally or for the requested deal. If an option is NULL it will not be changed.

Since

0.9.0

Availability ALL

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*LogAction*	Determines how messages are returned. Must be one of the following: • LOG_OPTION_SUPPRESS • LOG_OPTION_POPUP • LOG_OPTION_OVERWRITE • LOG_OPTION_APPEND • LOG_OPTION_OVERWRITE_POPUP • LOG_OPTION_APPEND_POPUP
in	*LogFile*	Fully qualified path and name of the log file.
in	*Debug*	Whether or not to produce detailed debug info. (irrelevant if LogAction = 0) • TRUE : Output detailed debug info. • FALSE : NOT output detailed debug info.

Return values

1	Unable to write to log file. No file logging (pop_up notification, if requested, will be provided).
0	No error
-1	Error - Invalid LogAction
-2	Error - Other error

Example:

```
*       CMO_STRUCT cmo;
*       memset(&cmo, 0, sizeof(CMO_STRUCT));
*
*       BOOLYAN Debug=1;
*       short action = LOG_OPTION_APPEND_POPUP;
*       set_log_options(NULL, &action, "log.txt", &Debug);
```

```
 *
 *       void* tid = NULL;
 *       int iret = create_deal_scenario_object(&tid,NULL,NULL,NULL);
 *       set_log_options(tid, &action, "log_1.txt", &Debug); //The log file must be unique
        for every deal/scenario object
 *
 *       open_deal_ex(tid,&cmo);
 *       close_deal_ex(tid,&cmo);
 *       release_deal_scenario_object(&tid);
 *
```

Warning

Different deal/scenario objects should use different LogFile.
Logging (esp with Debug set to TRUE) should be enabled for debugging purpose only. Detailed pay rule information, cashflow output, and other debugging information will fill up disk space quickly.

See Also

create_deal_scenario_object()

22.1.4.100 int set_maximum_deal_scenario_objects (int *max*)

Warning

This method is obsolete and does *NOT* perform anything when it is called.

Deprecated This method is deprecated and does *NOT* perform anything when it is called.

22.1.4.101 void CHASAPI set_missing_interest_rates_handling (MISSING_INTEREST_RATES_HANDLING *handling*)

This function sets the rule to handle the case where some interest rates are missing.

Since

0.9.0

Availability ALL

Parameters

in	*handling*	one of MISSING_INTEREST_RATES_HANDLING

Returns

None

Example:

```
 *      set_missing_interest_rates_handling(
       MISSING_INTEREST_RATES_TREAT_AS_ERROR);
 *
```

22.1.4.102 int CHASAPI set_pool_level_user_data_for_cb (void * *tid*, const char * *reremic_deal_id_or_null*, short *loan_num*, void * *user_data*)

Registers pool level user data with the WSA API which will can be got from MARKIT_POOL_INFO.usr_data.

Since

1.2.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	0 for parent deal or name of the child deal.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*user_data*	Pointer to user data.

Return values

0	No error
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* pDeal=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(pCmo.dealid, "BAFC08R2");
*
*       open_deal_ex(pDeal, &deal);
*       // Deal is already opened
*
*       // set user data for collateral 1
*       short loan_num = 1;
*       int ret = set_pool_level_user_data_for_cb(pDeal, NULL, loan_num, (
      void*)"user data what passed in");
*
*       void* coll_it = obtain_collat_iterator(pDeal, 0);
*       assert(NULL != coll_it);
*       MARKIT_POOL_INFO* coll_info = NULL;
*       // loop all collaterals and get usr_data of collateral 1
*       while((coll_info = get_next_collat(pDeal, coll_it)))
*       {
*           if(loan_num == coll_info->loan_number)
*           {
*               char *szUsrData = (char *)coll_info->usr_data;
*               // the content of szUsrData should be "user data what passed in"
*               assert(0 == strcmp("user data what passed in", szUsrData);
*               break;
*           }
*       }
*
*       close_deal_ex(pDeal, &deal);
*
```

Note

- User is responsible for maintaining the data, the WSA API just passes the pointer along.

22.1.4.103 long CHASAPI set_prepayments_ex (void ∗ *tid,* short *type,* short *is_vector,* double ∗ *pval,* long *loan_num,* BOOLYAN *set_sup_remic*)

Sets the constant or vectored prepayment speed. It will be used for the pool specified by loan_num with the ability to apply to underlying deals if it is a reremic.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
`in`	*type*	Type of prepayment curve. Must be one of: • PREPAY_CURVE_PSA - The PSA Standard Prepayment Assumptions rate specifies an annual prepayment percentage as a function of the seasoning of the mortgages/loans. 100% PSA assumes prepayment rates of 0.2% CPR in the first month following origination of the mortgage loans and an additional 0.2% CPR in each succeeding month until the 30th month. In the 30th month and beyond, 100% PSA assumes a fixed annual prepayment rate of 6% CPR. Any PSA other than 100% is calculated by multiplying the annual prepayment rate (the CPR) by that multiple of 100%. • PREPAY_CURVE_SMM - The Single Monthly Mortality rate is the percentage of the mortgages/loans outstanding at the beginning of the month assumed to terminate during the month. It is mathematically related to CPR: (1 – SMM)^12 = 1 – CPR. Seasoning of mortgages/loans will not be accounted for if a vectored prepayment rate is set. • PREPAY_CURVE_CPR - The Constant Prepayment Rate is the percentage of the mortgages/loans outstanding at the beginning of a year assumed to terminate during the year. It is mathematically related to SMM: (1 – SMM)^12 = 1 – CPR. Seasoning of mortgages/loans will not be accounted for if a vectored prepayment rate is set. • PREPAY_CURVE_CPY - The Constant Prepayment Yield rate is equivalent to the Constant Prepayment Rate(CPR) except that it assumes prepayment only happens after contractual lockout and yield maintenance period. • PREPAY_CURVE_HEP - The Home Equity Prepayment rate is a measure of prepayments for closed-end, fixed rate HEL loans. The HEP curve is steeply ramped for the first 10 months, and then plateaus a fixed rate for the life of the loan. For example, a 24% HEP means the loan will start at a prepayment speed of 2.4% CPR and grow to 24% CPR over 10 months, staying fixed at 24% CPR for the remainder of the loan. • PREPAY_CURVE_ABS - Asset-Backed Securities(ABS): The Asset--Backed Securities rate defines an increasing sequence of monthly prepayment rates which correspond to a constant absolute level of loan prepayments in all future periods. It is mathematically related to SMM: SMM = ABS / (1 – ABS ∗ (Seasoning – 1)). • PREPAY_CURVE_CUS • PREPAY_CURVE_CPB
`in`	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
`in`	*pval*	A pointer to the prepayment speeds (or speed). Value for current period (0-indexed element) will not be applied.

in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will replace any specified underlying deal settings. If FALSE, this will NOT replace any underlying deal settings.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*      void* ptid=NULL;
*      CMO_STRUCT deal;
*      memset(&deal, 0, sizeof(CMO_STRUCT));
*      strcpy(deal.dealid,"BAFC08R2");
*
*      open_deal_ex(ptid,&deal);
*      // Deal is already opened
*
*      double rate;    // prepayment speed
*      short type;     // prepayment type
*
*      // Set 8% CPR for whole deal,
*      // And apply to underlying
*      type = PREPAY_CURVE_CPR;
*      rate = .08;
*      set_prepayments_ex(ptid, type, 0, &rate, -1, true);
*
*      // Set vector of SMM for first piece of collateral
*      // Do not apply to underlying deals.
*      // rates is a previously set vector with 300 entries.
*      set_prepayments_ex(ptid, PREPAY_CURVE_SMM, 300, rates, 1, false);
*
*      run_deal_ex(ptid,&deal);
*      close_deal_ex(ptid,&deal);
*
```

Note

- Prepayment speeds are expressed as decimals. 5.25% would be .0525.
- If loan_num = -1, the setting will apply to all collateral loans. If loan_num is 1-based index, the setting will only apply to the loan specified by loan_num.
- value for current period (0-indexed element) will not be applied.

See Also

- set_prepayments_ex can be used to set ppy rates for both top level deals and underlying deals.
- set_reremic_prepayments is used to set ppy rates for underlying deals. If the deal specified by para dealid is not an underlying deal, it will return an error.

22.1.4.104 long CHASAPI set_rate_ex (void ∗ *tid,* short ∗ *idx,* short *vector,* double ∗ *pval*)

Sets the constant or vector interest rate that will be used for the specified index.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*idx*	A pointer to the index to set. idx must be one of: • enums of INDEX_TYPE (In indextypes.h). • enums of INDEX_TYPE_EX (SFW and CDOnet deals). • index codes returned by get_required_rate_codes().
in	*vector*	The length of the vector pointed to by pval, or 0 if pval points to a constant.
in	*pval*	A pointer to the new rate value or values. Value for current period (0-indexed element) will not be applied.

Return values

0	No error
-1	Error - Deal not open
-3	Error - Invalid market index
-4	Error - No value passed or the value of vector is negative.
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* tid=NULL;
*       CMO_STRUCT pCmo;
*       memset(&pCmo, 0, sizeof(CMO_STRUCT));
*       strcpy(pCmo.dealid,"BAFC08R2");
*
*       open_deal_ex(tid,&pCmo);
*       // Deal is already opened
*
*       short type;
*       double rate;
*       short mktRates[MAX_INDEX_TYPES_EX] = {0};
*       // Determine which market rates are required and set them
*       int IR = get_required_rate_codes(tid, mktRates,
      MAX_INDEX_TYPES_EX);
*
*       // Set the market rate for LIBOR_1 as a constant if it is required
*       if(1 == mktRates[LIBOR_1])
*       {
*           type = LIBOR_1;
*           rate = .0525;                       //5.25%
*           set_rate_ex(tid, &type, 0, &rate);
*       }
*
*       // Set the market rate for LIBOR_6 as a vector if it is required
*       // rateVec is a vector containing the market rates
*       // rateVecSize is the number of rates in the vector to use
*       if(1 == mktRates[LIBOR_6])
*       {
*           type = LIBOR_6;
*           double rateVec[] ={0,0.0436,0.0426,0.0425,0.0211,0.024,0.033,0.0124,0.0116};
*           short rateVecSize = sizeof(rateVec)/sizeof(rateVec[0]);
*           set_rate_ex(tid, &type, rateVecSize, rateVec);
*       }
*
*       close_deal_ex_(tid,&pCmo);
*
```

Note

- The rates are expressed as a decimal: 5.25% would be .0525.
- The required rates for a deal can be determined by calling get_required_rate_codes().
- value for current period (0-indexed element) will not be applied.

- Index rates vector apply from the latest update date closest/relative to the settlement date. A floater bond will use period 1 rate assumption at first reset date since deal update date.
- Index rates vector is a monthly rate vector, regardless payment frequency of deal.
- When API run with predefined scenario in PA/MPA, user can set spot rate by set_rate_ex(), API will project rate vector by adding MPA/PA's future rate shifts to the spot rate.

22.1.4.105 long CHASAPI set_recoveries_ex (void ∗ *tid,* short *is_vector,* double ∗ *pval,* long *loan_num,* BOOLYAN *set_sup_remic*)

Sets the constant or vectored recovery rate that will be used for the pool specified by loan_num , with the option to apply to underlying deals if the deal is a reremic. The recovery rate is the percentage of total liquidations that will be returned to the CMO, less any amount that is used to reimburse the servicer for principal advances.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the recovery rates (or rate). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE, this will be applied to underlying deals. If FALSE, this will be NOT applied to underlying deals.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* tid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(tid,&deal);
*       // Deal is already opened
*
*       // Set the recovery rate for all collateral in the deal to 25%.
*       // Replace any underlying deal recoveries.
*       double rate = .25;
*       int IR = set_recoveries_ex(tid, 0, &rate, -1, true);
*
*       // Set the recovery rate for the first piece of collateral to a vector.
*       // Use 100 values (rateVec1 is an existing vector).
*       // Do NOT overwrite values for underlying deals
*       double rateVec1[100] = {0.01};
*       IR = set_recoveries_ex(tid, 100, rateVec1, 1, false);
*
*       run_deal_ex(tid,&deal);
*       close_deal_ex(tid,&deal);
*
```

Note

- If recovery lag is not 0, in case of default, the loss will be realized at the time of recovery, not the time of default. By default, the recovery and loss will be calculated using the rate at the time of recovery (see set_recovery_from). If recovery is 0, the recovery and loss will happen immediately when the default happens.
- If loan_num = -1, the setting will apply to all collateral loans. If loan_num is 1-based index, the setting will only apply to the loan specified by loan_num.
- Recovery rates are expressed as decimals. 5.25% would be .0525.
- value for current period (0-indexed element) will not be applied.

See Also

- set_recoveries_ex() can be used for both top level deals and underlying deals.
- set_reremic_recoveries() is used for underlying deal. If the deal specified by parameter dealid is not an underlying deal, it will return an error.
- set_recovery_from() can be used to specify whether to use the recovery rate at the time of default or at the time of recovery (in the case of non-zero lag) for SFW deals.

22.1.4.106 long CHASAPI set_recovery_lag_ex (void ∗ *tid,* short *val,* long *loan_num,* BOOLYAN *set_sup_remic*)

Sets the recovery lag (months from default to liquidation). The settings can be applied to underlying deals optionally if they are reremic.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*val*	The lag in months.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE, this will be applied to underlying deals; otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened
*
*       // Set the recovery lag (months between default and recovery) to
*       // 2 months for all collateral in the deal and the underlying
*       // deals
*       set_recovery_lag_ex(ptid, 2, -1, true);
*
*       // Set the recovery lag to 3 months for the first piece of
*       // collateral.
*       set_recovery_lag_ex(ptid, 3, 1, true);
*
*       close_deal_ex(ptid,&deal);
*
```

Note

If loan_num = -1, the setting will apply to all collateral loans. If loan_num is 1-based index, the setting will only apply to the loan specified by loan_num.

See Also

- set_recovery_lag_ex can be used for both top level deals and underlying deals.
- set_reremic_recovery_lag is used for underlying deal. If the deal specified by parameter dealid is not an underlying deal, it will return an error.

22.1.4.107 long CHASAPI set_reremic_default_from (void ∗ *tid*, char ∗ *dealid*, const short *type*)

Sets which balance defaults are calculated from for the underlying deal.

Since

1.1.0

Availability CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*type*	The type of balance the defaults are calculated from. Must be one of: • DEFLT_FROM_CURBAL: The current balance for the period. • DEFLT_FROM_ORIGBAL: The balance in period 0 (deal "as of" date). • DEFLT_FROM_ZERO_CPR: The scheduled balance if no prepayments and no defaults.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-10	Error - Invalid type.
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void * tid = NULL;
*       // Deal is already opened
```

```
*
*       // Set the defaults from the current balance for underlying deal
*       // 3443Z
*       set_reremic_default_from(tid, "3443Z",
      DEFLT_FROM_CURBAL);
*
```

22.1.4.108 long CHASAPI set_reremic_defaults (void ∗ *tid,* char ∗ *dealid,* short *type,* short *is_vector,* double ∗ *pval,* long *loan_num*)

Sets the constant or vectored default rate that will be used for the pool specified by loan_num.

Since

0.9.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*type*	Type of default curve. Must be one of: • DEFAULT_CURVE_CDR - Constant Default Rate(CDR): Default percentage expressed as an annual compounded rate. • DEFAULT_CURVE_SDA - Standard default curve, measuring for defaults in the residential mortgage market. • DEFAULT_CURVE_MDR - Monthly Default Rate. • DEFAULT_CURVE_PCT • DEFAULT_CURVE_SEASONED_CDR • DEFAULT_CURVE_SEASONED_MDR

in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the default rates (or rate). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the collateral or -1 to apply to all collateral in the deal.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-3	Error - Invalid collateral index
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened
*
*       int remic_num = view_reremic_deals(ptid,NULL,NULL);
*       if(remic_num>0)
*       {
*           std::vector<CMO_STRUCT>remics(remic_num);
*           remic_num = view_reremic_deals(ptid,NULL,&remics.front());
*           double def = .01;
*           double def_array[] = {.01,.02,.03,.04,.05};
*           for(int i = 0;i<remic_num; ++i)
*           {
*             //set constant def rate for all collaterals in the underlying deal
*             set_reremic_defaults(ptid, remics[i].dealid,
      DEFAULT_CURVE_CDR, 0, &def, -1);
*
*             //set default rates vector for loan 1 in the underlying deal
*             set_reremic_defaults(ptid, remics[i].dealid,
      DEFAULT_CURVE_CDR, 5, def_array,1);
*           }
*       }
*
*       run_deal_ex(ptid,&deal);
*       close_deal_ex(ptid,&deal);
*
```

Note

- Default rates are expressed as decimals. 5.25% would be .0525.
- If loan_num = -1, the setting will apply to all collateral loans. If loan_num is 1-based index, the setting will only apply to the loan specified by loan_num.
- value for current period (0-indexed element) will not be applied.

See Also

- set_defaults_ex can be used to set default rates for both top level deals and underlying deals.
- set_reremic_defaults is used to set default rates for underlying deal. If the deal specified by para dealid is not an underlying deal, it will return an error.

22.1.4.109 long CHASAPI set_reremic_prepayments (void ∗ *tid,* char ∗ *dealid,* short *type,* short *is_vector,* double ∗ *pval,* long *loan_num*)

Sets the constant or vectored prepayment speed that will be used for the collateral specified by loan_num.

Since

0.9.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
`in`	*dealid*	The name of the underlying deal.
`in`	*type*	Type of prepayment curve. Must be one of: • PREPAY_CURVE_PSA - Standard prepayment curve measuring for prepayments in the residential mortgage market. • PREPAY_CURVE_SMM - Monthly prepayment or default rate. • PREPAY_CURVE_CPR - Constant Prepayment Rate(CPR): Prepayment percentage expressed as an annual compounded rate. • PREPAY_CURVE_CPY - Constant Prepayment Yield(CPY): It is equivalent to the Constant Prepayment Rate(CPR) except that it assumes prepayment only happens after contractual lockout and yield maintenance period. • PREPAY_CURVE_HEP - Home Equity Prepayment: A measure of prepayments for closed-end, fixed rate HEL loans. This curve accounts for the faster seasoning ramp for home equity loans. • PREPAY_CURVE_ABS - Asset-Backed Securities(ABS): It is used in ABS markets, where prepayments differ significantly from standard mortgages. This model defines an increasing sequence of monthly prepayment rates, which correspond to a constant absolute level of loan prepayments in all future periods. • PREPAY_CURVE_CUS • PREPAY_CURVE_CPB

in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the prepayment speeds (or speed). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-3	Error - Invalid collateral index
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened
*
*       int remic_num = view_reremic_deals(ptid,NULL,NULL);
*       if(remic_num>0)
*       {
*           std::vector<CMO_STRUCT>remics(remic_num);
*           remic_num = view_reremic_deals(ptid,NULL,&remics.front());
*           double ppy = .01;
*           double ppy_array[] = {.01,.02,.03,.04,.05};
*           for(int i = 0;i<remic_num; ++i)
*           {
*               //set constant ppy rate for all collaterals in the underlying deal
*               set_reremic_prepayments(ptid, remics[i].dealid,
       PREPAY_CURVE_CPR, 0, &ppy, -1);
*
*               //set ppy rates vector for loan 1 in the underlying deal
*               set_reremic_prepayments(ptid, remics[i].dealid,
       PREPAY_CURVE_CPR, 5, ppy_array,1);
*           }
*       }
*
*        close_deal_ex(ptid, &deal);
*
```

Note

- Prepayment speeds are expressed as decimals. 5.25% would be .0525.
- If loan_num = -1, the setting will apply to all collateral loans. If loan_num is 1-based index, the setting will only apply to the loan specified by loan_num.
- If loan_num >=1 and is invalid, it will return error code -3.
- value for current period (0-indexed element) will not be applied.

See Also

- set_prepayments_ex can be used to set ppy rates for both top level deals and underlying deals.
- set_reremic_prepayments is used to set ppy rates for underlying deal. If the deal specified by parameter dealid is not an underlying deal, it will return an error.

22.1.4.110 long CHASAPI set_reremic_recoveries (void ∗ *tid,* char ∗ *dealid,* short *is_vector,* double ∗ *pval,* long *loan_num*)

Sets the constant or vectored recovery rate in the underlying deal for the pool specified by loan_num. The recovery rate is the percentage of total liquidations that will be returned to the CMO, minus any amount that is used to reimburse the servicer for principal advances.

Since

0.9.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the recovery rates (or rate). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the collateral or -1 to apply to all collateral in the deal.

Return values

0	No error
-1	Error Deal not opened
-2	Error Requested underlying deal not part of the deal
-3	Error Invalid collateral index
-99	Error Invalid dso identifier (tid) or other errors, please see details by calling get_-deal_error_msg()

Note

- Recovery rates are expressed as decimals. 5.25% would be .0525.
- If loan_num = -1, the setting will apply to all collateral loans. If loan_num is 1-based index, the setting will only apply to the loan specified by loan_num.

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened
*
*       int remic_num = view_reremic_deals(ptid,NULL,NULL);
*       if(remic_num>0)
*       {
*           std::vector<CMO_STRUCT>remics(remic_num);
*           remic_num = view_reremic_deals(ptid,NULL,&remics.front());
*           double recovery = .01;
*           double recovery_array[] = {.01,.02,.03,.04,.05};
*           for(int i = 0;i<remic_num;++i)
*           {
*             // set constant recovery rate for all collaterals in the underlying deal.
*             set_reremic_recoveries(ptid, remics[i].dealid, 0, &recovery, -1);
*
*             // set recovery rates vector for loan 1 in the underlying deal.
*             set_reremic_recoveries(ptid, remic[i].dealid, 5, recovery_array, 1);
*           }
*       }
*
*       run_deal_ex(ptid,&deal);
*       close_deal_ex(ptid,&deal);
*
```

Note

- value for current period (0-indexed element) will not be applied.

See Also

- set_recoveries_ex can be used for both top level deals and underlying deals.
- set_reremic_recoveries is used for underlying deals. If the deal specified by para dealid is not an underlying deal, it will return an error.

22.1.4.111 long CHASAPI set_reremic_recovery_lag (void ∗ *tid,* char ∗ *dealid,* short *val,* long *loan_num*)

Sets the recovery lag (months from default to liquidation) in the underlying deal for either the collateral specified by loan_num or for the deal as a whole if loan_num = -1.

Since

0.9.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*val*	The lag in months.
in	*loan_num*	The 1-based index of the collateral or -1 to apply to all collateral in the deal.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-3	Error - Invalid collateral index
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened
*
*       int remic_num = view_reremic_deals(ptid,NULL,NULL);
*       if(remic_num>0)
*       {
*           std::vector<CMO_STRUCT>remics(remic_num);
*           remic_num = view_reremic_deals(ptid,NULL,&remics.front());
*           for(int i=0; i<remic_num;++i)
*           {
*               //set recovery lag to 2 months for all collaterals in the underlying deal
*               set_reremic_recovery_lag(ptid, remics[i].dealid, 2, -1);
*
*               // set recovery lag to 3 months for loan 1 in the underlying deal
*               set_reremic_recovery_lag(ptid, remics[i].dealid, 3, 1);
*           }
*       }
*
*       run_deal_ex(ptid,&deal);
*       close_deal_ex(ptid,&deal);
*
```

See Also

- set_recovery_lag_ex can be used for both top level deals and underlying deals.
- set_reremic_recovery_lag is used for underlying deal. If the deal specified by parameter dealid is not an underlying deal, it will return an error.

22.1.4.112 long CHASAPI set_reremic_service_advances (void ∗ *tid,* char ∗ *dealid,* const int *type*)

Sets the type of cashflows the servicer advances when the underlying deal's collateral is delinquent or in default.

Since

0.9.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*type*	The types of cashflow advanced. Must be one of the following: • SERVICER_ADVANCES_NOTHING • SERVICER_ADVANCES_INTEREST • SERVICER_ADVANCES_BOTH

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-10	Error - Invalid type
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "BAFC08R2";
*       int len = sizeof(deal_id) / sizeof(char);
*       memcpy(pCmo->dealid,deal_id,len);
*
*       open_deal_ex(tid,pCmo);
*
*       set_reremic_service_advances(tid, "nullified dealid",
       SERVICER_ADVANCES_BOTH);
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.113 int CHASAPI set_reremic_trigger_override (void ∗ *tid,* char ∗ *dealid,* char ∗ *trigger_name,* short *is_vector,* SBYTE ∗ *override*)

Overrides the trigger calculations for the indicated trigger in the underlying deal.

Since

1.2.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*trigger_name*	The case-sensitive name of the trigger whose information is being overridden.
in	*is_vector*	The entries in the vector being passed, or -1 if a constant is passed. The first entry in a vector corresponds to period 0.
in	*override*	A pointer to the value(values) being passed. Each value is a signed char(SBYTE) with one of the following values: • -1: Ignore override – use the paydown rules calculation • 0: Trigger fails – (condition not met). • 1: Trigger passes – (condition met).

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-3	Error - Trigger not in deal
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Example:

```
*      void *tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(CMO_STRUCT));
*      char deal_id[] = "BAFC08R2";
*      int len = sizeof(deal_id) / sizeof(char);
*      memcpy(pCmo->dealid,deal_id,len);
*
*      open_deal_ex(tid,pCmo);
*
*      int remic_num = view_reremic_deals(tid, NULL, NULL);
*      std::vector<CMO_STRUCT> remics(remic_num);
*
*      int trg_num = get_reremic_triggers_avail(tid, remics[0].dealid, NULL, NULL
     );
*      if(trg_num>0)
*      {
*          std::vector<char> name_buf(trg_num*21);
*          std::vector<char*> names(trg_num);
*          for(int i = 0; i<trg_num; i++ )
*          {
*              names[i] = &name_buf[i*21];
*          }
*          get_reremic_triggers_avail(tid, remics[0].dealid, &names.front(), NULL
     );
*
*          signed char over_ride = 1;
*          for(int i = 0; i<trg_num; i++)
*          {
*              set_reremic_trigger_override(tid, remics[0].dealid, names[i], 0,
       &over_ride);
*          }
*      }
*
*      close_deal_ex(tid,pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.1.4.114 void CHASAPI set_service_advances_ex (void * *tid*, const int *type*, BOOLYAN *set_sup_remic*)

Sets the type of cashflows the servicer advances when the collateral is in default, with the option to apply the setting to underlying deals if they are a reremic.

Since

1.1.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	The types of cashflow advanced. Must be one of the following: • SERVICER_ADVANCES_NOTHING - The servicer advances neither interest nor principal. • SERVICER_ADVANCES_INTEREST - The servicer advances interest only. • SERVICER_ADVANCES_BOTH - The servicer advances both interest and principal.
in	*set_sup_remic*	If TRUE this will be applied to underlying deals. If FALSE, then the underlying deals will use the system default, SERVICER_ADVANCES_NOTHING.

Returns

None

Example:

```
*       void* ptid=NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       open_deal_ex(ptid,&deal);
*       // Deal is already opened
*
*       // Set the servicer to advance both interest and principal
*       // Apply to any underlying deals
*       set_service_advances_ex(ptid,
       SERVICER_ADVANCES_BOTH, true);
*
*       run_deal_ex(ptid,&deal);
*       close_deal_ex(ptid,&deal);
*
```

Note

If this function is not called, the system default is to use SERVICER_ADVANCES_NOTHING.

See Also

- This function replaces set_services_advances, which has been deprecated.
- set_service_reimburse_advint, which can be used to set whether the servicer will be reimbursed for principal and interest advances.
- set_reremic_serice_advances() sets the type of cashflow that the servicer advances when the underlying deal's collateral is delinquent or in default.

22.1.4.115 void CHASAPI set_tmp_dir_treatment (**TMP_DIR_TREATMENT** *tmp_dir_treatment*)

Sets the temporary files clean up method when the API exits.

Since

1.5.0

Availability ALL

Precondition

None. It can be called anywhere before the API exits.

Parameters

in	*tmp_dir_-treatment*	Must be enum of TMP_DIR_TREATMENT

Returns

None

Example:

```
*      // Try to remove all temporary files if possible when it exits
*      set_tmp_dir_treatment(TMP_DIR_REMOVE_ALL);
*
```

Note

The API uses advisory file locks. The current implement for cleanup procedure will NOT remove any temporary files when it exits if there are other WSA API processes holding the file locks.

22.1.4.116 int CHASAPI set_trigger_override (void ∗ *tid,* char ∗ *trigger_name,* short *is_vector,* **SBYTE** ∗ *override*)

Overrides the trigger calculations for the indicated trigger.

Since

1.2.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*trigger_name*	The case-sensitive name of the trigger whose information is being overridden.
in	*is_vector*	The entries in the vector being passed, or -1 if a constant is passed. The first entry in a vector corresponds to period 0.
in	*override*	A pointer to the value(values) being passed. Each value is a signed char(SB-YTE) with one of the following values: • -1 Ignore override – use the paydown rules calculation • 0 Trigger fails – (condition not met). • 1 Trigger passes – (condition met).

Return values

0	No error
-1	Error - Deal not opened
-3	Error - Trigger not in deal
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "AL2010-A";
*       int len = sizeof(deal_id) / sizeof(char);
*       memcpy(pCmo->dealid,deal_id,len);
*
*       open_deal_ex(tid,pCmo);
*
*       int trg_num = get_triggers_avail(tid, NULL, NULL);
*       if(trg_num>0)
*       {
*           std::vector<char> name_buf(trg_num*21);
*           std::vector<char*> names(trg_num);
*           for(int i = 0; i<trg_num; i++ )
*           {
*               names[i] = &name_buf[i*21];
*           }
*           get_triggers_avail(tid, &names.front(), NULL);
*           signed char over_ride = 1;
*
*           for(int i = 0; i<trg_num; i++)
*           {
*               set_trigger_override(tid, names[i], 0, &over_ride);
*           }
*       }
*
*       run_deal_ex(tid,pCmo);
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.117 int CHASAPI set_user_data_for_cb (void ∗ *tid,* void ∗ *user_data*)

Registers user data with the WSA API which will then be passed to call back functions. This allows user to maintain the state between calls to call back functions.

Since

1.2.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*user_data*	Pointer to user data

Return values

0	No error
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Note

1. User data is stored for a given tid and is passed back to all call back function.
2. User is responsible for maintaining the data, the WSA API just passes the pointer along.
3. User might request an automatic clean up function to be called by the WSA API (see reference to install_user_cleanup_cb())

Example:

```
*
*     int fCOLLAT_ASSUMP_CB1(void* tid,
*                          char* first_period_date,
*                          int max_periods,
*                          PAY_POOL_INFO* pool_info,
*                          CCMO_COLLAT_ASSUMPTIONS* assumptions,
*                          void* user_data,
*                          char* error_message,
*                          int max_size_of_error_message
*                          );
*
*      void *tid = NULL;
*      CMO_STRUCT deal;
*      memset(&deal, 0, sizeof(CMO_STRUCT));
*      strcpy(deal.dealid,"AL2010-A");
*
*      create_deal_scenario_object(&tid,NULL,NULL,NULL);
*      open_deal_ex(tid,&deal);
*
*      set_user_data_for_cb(tid,(void*)"set_user_data_for_cb");
*
*      install_collat_assump_cb(tid,fCOLLAT_ASSUMP_CB1);
*
*      run_deal_ex(tid,&deal);
*      close_deal_ex(tid,&deal);
*      release_deal_scenario_object(&tid);
*
```

22.1.4.118 long CHASAPI view_all_bonds_ex (void * *tid,* CCMO_BONDS_S *all_bonds[ ]*)

Populates an array of bond structures (CCMO_BONDS_S) with descriptive information on all bonds in the currently open deal.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*all_bonds*	An allocated array of CCMO_BONDS_S structures.

Return values

0	No error
-1	Error - Deal not open
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Note

The array all_bonds[] must be allocated to be of at least as long as the number of bonds in the deal. The number of bonds is set by open_deal_ex() and is found in the CMO_STRUCT num_bonds member.

Example:

```
*      void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       open_deal_ex(tid,pCmo);
*
*       std::vector<CCMO_BONDS_S> pBonds(pCmo->num_bonds+1);
*       view_all_bonds_ex(tid,&pBonds.front());
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.1.4.119 int CHASAPI view_bond_coll_groups (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *bondid,* int *groups_array[ ],* int *groups_array_length,* int ∗ *total_groups*)

Returns the number of related poolgroups of the specified bond and populates an array of collateral group IDs related to the specified bond.

Since

1.5.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	Reremic deal id for a child deal, otherwise null.
in	*groups_array_-length*	Size of array groups_array
in	*bondid*	A pointer to the name of the bond.
out	*groups_array*	A pointer to the array of collateral group IDs related to the specified bond.
out	*total_groups*	The number of groups related to the specified bond.

Returns

The actual number of groups returned in the array.

Example:

```
*    void *tid = NULL;
*    CMO_STRUCT *pCmo = new CMO_STRUCT();
*    memset(pCmo, 0, sizeof(CMO_STRUCT));
*
*    set_input_path("C:\\");
*    if(open_deal_ex(tid, pCmo)!=0)
*    {
*        delete pCmo;
*        return;
*    }
*
*    int groups[100]={0};
*    int totalgroups=0;
*    int num_group_ret=view_bond_coll_groups(tid, NULL, "A1", groups, 100, &totalgroups
       ));
*    for(int i=0;i<totalgroups; i++)
*    {
*        //do what you want to do with groups[i]
*    }
*
*    close_deal_ex(tid, pCmo);
*    delete pCmo;
*    pCmo = NULL;
*
```

Note

total_groups and the return value can be different if:

- The array provided is smaller than the actual number of pool groups supporting the bond. In this case, total_groups might be bigger than the return value.
- For SFW deal, for some given pool groups, there are several smaller 'virtual' pool groups that belong to those groups. In this case, total_groups (representing the number of larger pool groups) might be smaller than the return value.

See Also

- view_coll_groups()
- get_average_collat_by_bond()

22.1.4.120 int CHASAPI view_coll_groups (void ∗ *tid,* int *groups_array[ ],* int *groups_array_length,* int ∗ *total_groups*)

This function will return the number of collateral groups in a deal and will populate the user provided array with group numbers.

Since

0.9.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier.
`in,out`	*groups_array*	A user allocated array to which collateral group numbers will be written.
`in`	*groups_array_-length*	The size of the groups_array. To make sure the API does not overrun user's memory.
`out`	*total_groups*	The WSA API will provide total group number in the deal. That might be different from the value returned by the function if user did not allocate enough space.

Return values

number	The number of groups returned by the function
<0	Error - Check error message

Example:

```
*       void * pDeal = NULL;
*       // deal has already opened.
*
*       int numGroups(0);
*       assert(0 == view_coll_groups(pDeal, NULL, NULL, &numGroups));  // get the total
        groups number
*
*       std::vector<int> groups(numGroups,0);
*       int total_groups = 0;
*       view_coll_groups(pDeal, &groups[0], numGroups, &total_groups);
*
```

Note

If groups_array is NULL and groups_array_length is 0, total groups number will be returned by total_groups parameter.

22.1.4.121 long CHASAPI view_colls_ex (void * *tid,* short *index,* CCMO_POOL_INFO *all_colls[ ],* CCMO_POOL_INFO_EX *all_colls_ex[ ],* short *pool_size,* short *pool_ex_size,* short *arm_size*)

Returns the pool information for either all collateral or the requested piece of collateral in a deal.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*index*	The 0-based index of the loan (-1 for all loans).
in	*pool_size*	The size of the CCMO_POOL_INFO structure.
in	*pool_ex_size*	The size of the CCMO_POOL_INFO_EX structure.
in	*arm_size*	The size of the CCMO_ARM_INFO structure. Pass 0 if you do not want arm information.
out	*all_colls*	A client-allocated array of CCMO_POOL_INFO structures.
out	*all_colls_ex*	A client-allocated array of CCMO_POOL_INFO_EX structures.

Return values

0	No error
-1	Error - Deal not opened
-3	Error - Invalid loan index
-4	Error - Invalid size
-5	Error - No output vector passed
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Note

If all collateral is requested the arrays all_colls[] and all_colls_ex must be allocated to be at least as long as the value CMO_STRUCT.num_colls returned by open_deal_ex(). If arm_info is requested the arm member of each CCMO_POOL_INFO_EX must be allocated.

Example:

```
*      void* ptid=NULL;
*      CMO_STRUCT deal;
*      memset(&deal, 0, sizeof(CMO_STRUCT));
*      strcpy(deal.dealid,"AL2010-A");
*
*      create_deal_scenario_object(&ptid,NULL,NULL,NULL);
*      open_deal_ex(ptid,&deal);
*
*      CCMO_POOL_INFO all_colls[100]={0};
*      CCMO_POOL_INFO_EX all_colls_ex[100]={0};
*      view_colls_ex(ptid,0,NULL,NULL,0,0,0);
*      view_colls_ex(ptid, 0, all_colls, all_colls_ex, sizeof(
      CCMO_POOL_INFO), sizeof(CCMO_POOL_INFO_EX), 1);
*
*      close_deal_ex(ptid,&deal);
*      release_deal_scenario_object(&ptid);
*
```

22.1.4.122 long CHASAPI view_reremic_colls_ex (void * *tid,* char * *dealid,* short *index,* CCMO_POOL_INFO *all_colls[ ],* CCMO_POOL_INFO_EX *all_colls_ex[ ],* short *pool_size,* short *pool_ex_size,* short *arm_size*)

Returns the pool information for either all collateral or the requested piece of collateral in an underlying deal.

Since

0.9.0

Availability CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal.
in	*index*	The 0-based index of the loan (-1 for all loans).
in	*pool_size*	The size of the CCMO_POOL_INFO structure.
in	*pool_ex_size*	The size of the CCMO_POOL_INFO_EX structure.
in	*arm_size*	The size of the ARM_INFO structure. Pass 0 if you do not want arm information.
out	*all_colls*	A client-allocated array of CCMO_POOL_INFO structures.
out	*all_colls_ex*	A client-allocated array of CCMO_POOL_INFO_EX structures. The *arm* structure must either be allocated (CCMO_ARM_INFO) or set to 0.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-3	Error - Invalid loan index
-4	Error - Invalid structure size
-5	Error - No output structure passed
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get_deal_error_msg()

Note

If all collateral is requested the arrays all_colls[] and all_colls_ex[] must be allocated to be of at least as long as the value num_colls returned by open_deal_ex() (obtained by view_reremic_deals()). If arm_info is requested, the arm member of each CCMO_POOL_INFO_EX must be allocated.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(CMO_STRUCT));
*       char deal_id[] = "BAFC08R2";
*       int len = sizeof(deal_id) / sizeof(char);
*       memcpy(pCmo->dealid,deal_id,len);
*
*       open_deal_ex(tid,pCmo);
*
*       CMO_STRUCT remics[10] = {0};
*       view_reremic_deals(tid,NULL,remics);
*
*       std::vector<CCMO_POOL_INFO> cpi(remics[0].num_colls);
*       std::vector<CCMO_POOL_INFO_EX> cpi_ex(remics[0].num_colls);
*       std::vector<CCMO_ARM_INFO> cpi_arm_info(remics[0].num_colls);
*       size_t NUM_COLLS = std::abs(remics[0].num_colls);
*       for (size_t i=0; i<NUM_COLLS; ++i)
*       {
*           cpi_ex[i].arm=&(cpi_arm_info[i]);
*       }
*
*       view_reremic_colls_ex(tid, remics[0].dealid, -1, &cpi.front(), &cpi_ex.front(),
       sizeof(CCMO_POOL_INFO),sizeof(CCMO_POOL_INFO_EX),sizeof(
       CCMO_ARM_INFO));
*
*       close_deal_ex(tid,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*       all_colls_info.clear();
*
```

22.1.4.123 int CHASAPI view_reremic_deals (void ∗ *tid*, char ∗ *dealid*, CMO_STRUCT *remics[]*)

Populates an array of CMO_STRUCT structures with descriptive information on all underlying deals for the currently open deal if it is a reremic. Please note that only info of opened underlying deals will be returned. The array remics[] must be allocated to be of at least as long as the number of underlying remics requested.

The return value is the number of underlying remics. To get the number of underlying deals, pass NULL for the remics[] parameter.

To get information for all underlying deals, pass NULL for the dealid. To get a single underlying deal info, pass the dealid for the underlying deal. Then the CMO_STRUCT will have information for that deal only.

Since

0.9.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*dealid*	The name of the underlying deal. To get information on all underlying deals, pass NULL.
out	*remics*	An array of CMO_STRUCT structures, which must be allocated to hold at least as many deals as requested (either one or all underlying remics, depending on the dealid parameter).

Return values

>=0	The number of underlying deals
-1	Error - Deal not opened
-2	Error - Requested underlying deal not part of the deal
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       // Deal is already opened
*
*       // Get the count and allocate the structures
*       int cDeals = view_reremic_deals(NULL, NULL, NULL);
*       CMO_STRUCT * allDeals = new CMO_STRUCT[cDeals];
*       // Try to get a specific underlying deal
*       view_reremic_deals(tid, "2563", allDeals);
*       // Get all of the underlying deals. Note that IR will be the same as cDeals
*       view_reremic_deals(tid, NULL, allDeals);
*
*       // Print the underlying deal names in a previously opened file
*       fprintf(fle, "Underlying deals for %s\n", pCmo->dealid);
*       for(idx = 0; idx < cDeals; idx++)
*          fprintf(fle, " %s", allDeals[idx].dealid);
*
```

22.1.4.124 void CHASAPI write_log (void ∗ *tid*, char ∗ *message*, int *log_level*)

This function writes the given message to the log file.

Since

1.1.0

Availability ALL

Parameters

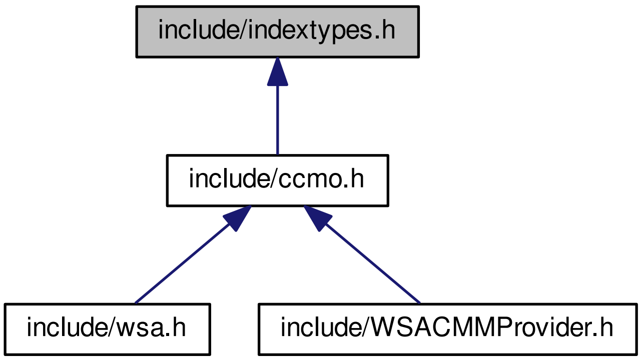
in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*message*	Message to be written
in	*log_level*	

Example:

```
    *       void *tid = NULL;
    *       char teststring[20] = "abcdefgh";
    *       write_log(tid, teststring, 1);
    *
```

22.2 include/indextypes.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum POOL_TYPE {
  FHLMC, GOLD, FNMA, GNMA1,
  GNMA2, WHOLE, ARM, REMIC,
  MIXED, SBA, STUD_LN, AUTO,
  AUTO_LS, HELOC, POOL_TYPE_OTHER, OTHER =POOL_TYPE_OTHER }
- enum INDEX_TYPE {
  NONE, LIBOR_1, LIBOR_3, LIBOR_6,
  LIBOR_12, LIBOR_24, LIBOR_36, LIBOR_48,
  LIBOR_60, LIBOR_84, LIBOR_120, LIBOR_240,
  LIBOR_360, TSY_1, TSY_3, TSY_6,
  TSY_12, TSY_24, TSY_36, TSY_48,
  TSY_60, TSY_84, TSY_120, TSY_240,
  TSY_360, COFI, PRIME, INDEX_TYPE_WAC,
  WAC =INDEX_TYPE_WAC, **MAX_INDEX_TYPES** }
- enum POOL_PURPOSE_TYPES {
  POOL_PURPOSE_OTHER, POOL_PURPOSE_PURCHASE, POOL_PURPOSE_CASHOUT_REFINANC-E, POOL_PURPOSE_HOME_IMPROVEMENT,
  POOL_PURPOSE_NEW_CONSTRUCTION, POOL_PURPOSE_RATE_TERM_REFINANCE, **POOL_PUR-POSE_NOT_AVAILABLE** }

- enum POOL_PROPERTY_TYPES {
POOL_PROPERTY_OTHER, POOL_PROPERTY_SINGLE_FAMILY, POOL_PROPERTY_MULTI_FAMILY, POOL_PROPERTY_CONDO,
POOL_PROPERTY_PUD, POOL_PROPERTY_COMMERCIAL, POOL_PROPERTY_COOP, POOL_PROPERTY_MOBILE_HOME,
POOL_PROPERTY_MANUFACTURED_HOUSING, **POOL_PROPERTY_NOT_AVAILABLE** }
- enum POOL_OCCUPANCY_TYPES {
POOL_OCCUPANCY_OTHER, POOL_OCCUPANCY_OWNER, POOL_OCCUPANCY_SECOND, POOL_OCCUPANCY_INVESTOR,
POOL_OCCUPANCY_NOT_AVAILABLE }
- enum POOL_DELINQ_STATES {
POOL_DELINQ_CURRENT, POOL_DELINQ_30P, POOL_DELINQ_60P, POOL_DELINQ_90P,
POOL_FORECLOSED, POOL_REO, POOL_TERMINATED, **POOL_DELINQ_STATES_SIZE** }
- enum POOL_DOCUM_TYPES {
POOL_DOCUM_OTHER, POOL_DOCUM_FULL, POOL_DOCUM_LIMITED, POOL_DOCUM_SISA,
POOL_DOCUM_SIVA, POOL_DOCUM_NINA, POOL_DOCUM_NO_RATIO, POOL_DOCUM_NO_DOC,
POOL_DOCUM_ALTERNATIVE, **POOL_DOCUM_UNKNOWN** =99 }
- enum POOL_LIEN_TYPES { POOL_LIEN_TYPE_OTHER, POOL_LIEN_TYPE_FIRST, POOL_LIEN_TYPE_SECOND, **POOL_LIEN_TYPE_THIRD** }
- enum POOL_FIELDS_LENGTH {
POOL_ID_LENGTH =20, POOL_SERVICER_NAME_LENGTH =11, POOL_STATE_LENGTH =3, POOL_COUNTRY_LENGTH = 4,
POOL_PURPOSE_LENGHT = 5, POOL_PROPERTY_TYPE_LENGTH =5, POOL_OCCUPANCY_LENGTH =5, POOL_ZIP_LENGTH =6,
POOL_CURRENCY_LENGTH = 4 }

  Enum for the length of some pool fields.
- enum GROUP_DELINQ_STATES {
GROUP_DELINQ_30, GROUP_DELINQ_60, GROUP_DELINQ_90, GROUP_DELINQ_120,
GROUP_DELINQ_150, GROUP_DELINQ_180, GROUP_DELINQ_210, GROUP_DELINQ_240,
GROUP_DELINQ_270, GROUP_DELINQ_300, GROUP_DELINQ_330, GROUP_DELINQ_360,
GROUP_DELINQ_SIZE }

  Enum of group delinquency states, used by set_addit_group_delinquencies()

22.2.1 Enumeration Type Documentation

22.2.1.1 enum GROUP_DELINQ_STATES

Enumerator

GROUP_DELINQ_30	Group delinquencies less than 30 days.
GROUP_DELINQ_60	Group delinquencies within 30 ∼ 59 days.
GROUP_DELINQ_90	Group delinquencies within 60 ∼ 89 days.
GROUP_DELINQ_120	Group delinquencies within 90 ∼ 119 days.
GROUP_DELINQ_150	Group delinquencies within 120 ∼ 149 days.
GROUP_DELINQ_180	Group delinquencies within 150 ∼ 179 days.
GROUP_DELINQ_210	Group delinquencies within 180 ∼ 209 days.
GROUP_DELINQ_240	Group delinquencies within 210 ∼ 239 days.
GROUP_DELINQ_270	Group delinquencies within 240 ∼ 269 days.
GROUP_DELINQ_300	Group delinquencies within 270 ∼ 299 days.
GROUP_DELINQ_330	Group delinquencies within 300 ∼ 329 days.
GROUP_DELINQ_360	Group delinquencies within 330 ∼ 359 days.
GROUP_DELINQ_SIZE	Size of group delinquency states.

22.2.1.2 enum INDEX_TYPE

Enum of index rate type, used by set_rate_ex()

See Also

- CCMO_ARM_INFO::index
- CCMO_POOL_INFO::arm_index

Enumerator

LIBOR_1 Libor 1 month.

LIBOR_3 Libor 3 months.

LIBOR_6 Libor 6 months.

LIBOR_12 Libor 1 year.

LIBOR_24 Libor 2 years.

LIBOR_36 Libor 3 years.

LIBOR_48 Libor 4 years.

LIBOR_60 Libor 5 years.

LIBOR_84 Libor 7 years.

LIBOR_120 Libor 10 years.

LIBOR_240 Libor 20 years.

LIBOR_360 Libor 30 years.

TSY_1 Treasury 1 month.

TSY_3 Treasury 3 months.

TSY_6 Treasury 6 months.

TSY_12 Treasury 1 year.

TSY_24 Treasury 2 years.

TSY_36 Treasury 3 years.

TSY_48 Treasury 4 years.

TSY_60 Treasury 5 years.

TSY_84 Treasury 7 years.

TSY_120 Treasury 10 years.

TSY_240 Treasury 20 years.

TSY_360 Treasury 30 years.

COFI Cost of funds index.

PRIME Prime interest rate.

INDEX_TYPE_WAC Weighted coupon.

WAC Weighted coupon.

22.2.1.3 enum POOL_DELINQ_STATES

Enum of loan delinquency states

See Also

- MARKIT_POOL_INFO::delinquency
- CCMO_POOL_INFO_EX::delinquency

Enumerator

POOL_DELINQ_CURRENT Delinquency current status.

POOL_DELINQ_30P Delinquency (30, 60] days; or non-performing for bank loans.

POOL_DELINQ_60P Delinquency (60, 90] days.

POOL_DELINQ_90P Delinquency (90, 120] days.

POOL_FORECLOSED Foreclosed.

POOL_REO Real estate owned.

POOL_TERMINATED Terminated, including paid off, repurchased, liquidated, closed etc.

22.2.1.4 enum POOL_DOCUM_TYPES

Enum of loan documen type

See Also

- MARKIT_POOL_INFO::doc

Enumerator

POOL_DOCUM_OTHER Other.

POOL_DOCUM_FULL Full document type.

POOL_DOCUM_LIMITED Limited.

POOL_DOCUM_SISA Stated income, stated assets type.

POOL_DOCUM_SIVA Stated income, verified assets type.

POOL_DOCUM_NINA No income, no asset type.

POOL_DOCUM_NO_RATIO No ratio type.

POOL_DOCUM_NO_DOC No doc type.

POOL_DOCUM_ALTERNATIVE Alternative document type.

22.2.1.5 enum POOL_FIELDS_LENGTH

Enumerator

POOL_ID_LENGTH Length of Pool ID.

POOL_SERVICER_NAME_LENGTH Length of servicer name.

POOL_STATE_LENGTH Length of state.

POOL_COUNTRY_LENGTH Length of country.

POOL_PURPOSE_LENGHT Length of purpose.

POOL_PROPERTY_TYPE_LENGTH Length of property type.

POOL_OCCUPANCY_LENGTH Length of occupancy.

POOL_ZIP_LENGTH Length of ZIP code.

22.2.1.6 enum POOL_LIEN_TYPES

Enum of loan line type

See Also

- MARKIT_POOL_INFO::lien_type

Enumerator

POOL_LIEN_TYPE_OTHER	Other.
POOL_LIEN_TYPE_FIRST	First lien.
POOL_LIEN_TYPE_SECOND	Second lien.

22.2.1.7 enum POOL_OCCUPANCY_TYPES

Enum of loan occupancy type

See Also

- MARKIT_POOL_INFO::occupancy
- CCMO_POOL_INFO_EX::occupancy

Enumerator

POOL_OCCUPANCY_OTHER	Other.
POOL_OCCUPANCY_OWNER	Primary.
POOL_OCCUPANCY_SECOND	Secondary.
POOL_OCCUPANCY_INVESTOR	Investment.

22.2.1.8 enum POOL_PROPERTY_TYPES

Enum of loan property type

See Also

- MARKIT_POOL_INFO::property_type
- CCMO_POOL_INFO_EX::property_type

Enumerator

POOL_PROPERTY_OTHER	Other property type.
POOL_PROPERTY_SINGLE_FAMILY	Single family property.
POOL_PROPERTY_MULTI_FAMILY	Multifamily property.
POOL_PROPERTY_CONDO	Condo property.
POOL_PROPERTY_PUD	Planned unit development.
POOL_PROPERTY_COMMERCIAL	Commercial property.
POOL_PROPERTY_COOP	Co-op property.
POOL_PROPERTY_MOBILE_HOME	Mobile home.
POOL_PROPERTY_MANUFACTURED_HOUSING	Manufactured housing.

22.2.1.9 enum POOL_PURPOSE_TYPES

Enum of loan purpose

See Also

- MARKIT_POOL_INFO::purpose
- CCMO_POOL_INFO_EX::purpose

Enumerator

POOL_PURPOSE_OTHER Other purpose.

POOL_PURPOSE_PURCHASE For purchase purpose.

POOL_PURPOSE_CASHOUT_REFINANCE For cash out purpose, including education or medical expense.

POOL_PURPOSE_HOME_IMPROVEMENT For home improvement purpose.

POOL_PURPOSE_NEW_CONSTRUCTION For new construction purpose.

POOL_PURPOSE_RATE_TERM_REFINANCE For refinance purpose.

22.2.1.10 enum POOL_TYPE

Can be used to indicate the pool types.

See Also

- CCMO_POOL_INFO::type
- MARKIT_POOL_INFO::type

Enumerator

FHLMC Mortgage-backed securities issued by Freddie Mac (The Federal Home Loan Mortgage Corporation).

GOLD Gold Participant Certificate Securities, mortgage-backed securities issued by Freddie Mac.

FNMA Mortgage-backed securities issued by Fannie Mae (The Federal National Mortgage Association).

GNMA1 Single-issuer pools mortgage-backed securities issued by Ginnie Mae (Government National Mortgage Association).

GNMA2 Multiple-issuer pools mortgage-backed securities issued by Ginnie Mae (Government National Mortgage Association).

WHOLE Whole loan.

ARM Adjustable-rate mortgage.

REMIC A tranche of a REMIC (Real Estate Mortgage Investment Conduit). The pool is a bond (of an underlying deal).

MIXED Mixed-use mortgage.

SBA Small business administration.

STUD_LN Student loan.

AUTO Auto loan.

AUTO_LS Auto leases.

HELOC Home equity line of credit.

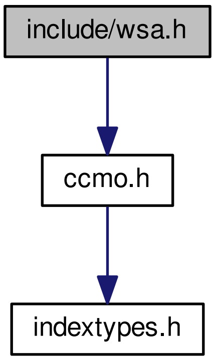
POOL_TYPE_OTHER All other pool types.

OTHER **Deprecated** Same as POOL_TYPE_OTHER

22.3 include/wsa.h File Reference

```
#include "ccmo.h"
```

Include dependency graph for wsa.h:



Data Structures

- struct DEAL_ACCOUNT_INFO

  This structure stores information about deal accounts (reserve, insurance, liquidity) in SFW deals.
- struct MOODYS_SSFA_CALC

  Simplified Supervisory Formula Approach (SSFA) calculation.
- struct MOODYS_BOND_HISTORY

  This structure stores historical information of a given bond.
- struct BOND_STEP_UP_COUPON

  This structure is passed to get coupon step up date and trigger.
- struct MOODYS_BOND_INFO

  Additional bond information.
- struct MOODYS_POOL_HISTORY

  This structure stores historical information of a given pool group.
- struct BORROWER_BENEFIT_ELIGIBILITY

  This structure describes borrower benefit for a student loan.
- struct MOODYS_STUDENT_LOAN_INFO

  This structure stores loan level information for a student loan.
- struct CDO_TEST_INFO

  This struct stores CDO test information.
- struct CDO_TEST_FLOW

  This struct stores the test projection information.
- struct CDO_DATE_INFO

  This struct stores the date info of the CDOnet Deal.
- struct GLOBAL_REINVESTMENT_INFO

  This struct stores information of global reinvestment setting.
- struct GLOBAL_REINVESTMENT_ASSET_INFO

  This struct stores asset information for global reinvestment.

- struct MONTE_CARLO_ASSUMPTION

  This struct stores settings of monte carlo assumption.
- struct MONTE_CARLO_DEF_PPY_REC_ASSUMPTION

  This struct stores settings of monte carlo economy assumption.
- struct MONTE_CARLO_RESULT

  This struct stores results of monte carlo run.
- struct FFIEC_INPUT_PARAMS

  This struct stores FFIEC tests input information.
- struct FFIEC_RESULTS

  This struct stores results for FFIEC test.
- struct DPD_ASSUMPTION

  Assumption for Default Probability Distribution.
- struct DPD_SCENARIO

  Scenario for Default Probability Distribution Simulation.
- struct DPD_RESULT

  Result for Default Probability Distribution Simulation.
- struct MOODYS_DEAL_INFO

  Additional deal information.
- struct COUPON_INFO

  coupon information.
- struct HECM_INFO
- struct WHOLE_LOAN_EXTEND_INFO
- struct BANKLOAN_CALL_ADJ_PARAM

  bankloan information.
- struct WHOLE_LOAN_SINK_FUND
- struct BANKLOAN_EXTEND_INFO
- struct UK_WHOLE_LOAN_INFO
- struct WHOLE_LOAN_STRUCT

  whole loan information.
- struct LOAN_PRICING_INPUT

  price_loan_ex input information.
- struct DISTRESSED_PROPERTY_RECOVERY

  distressed property recovery information.
- struct MOODYS_LOAN_CASHFLOW

  This structure is used to provide user with loan cashflow. It includes number of cashflow points as well as loan cashflow dates.
- struct MOODYS_ACCOUNT_CASHFLOW

  This structure is used to provide user with loan cashflow. It includes number of cashflow points as well as loan cashflow dates.
- struct MOODYS_HEDGE_OVERRIDE

  This structure stores hedge override information.
- struct MOODYS_HEDGE_STRUCT

  This structure stores hedge information.
- struct MOODYS_FEE_STRUCT

  This structure stores fee information.
- struct OAS_PRICE_STRUCT

  This structure stores OAS price array.
- struct METRIC_RESULTS_STRUCT_EX

  This structure stores metircs results information.
- struct METRIC_INPUT_STRUCT_EX

  This structure stores inputs information for metrics ex calculation.

- struct METRIC_INPUT_STRUCT

  This structure stores inputs information for metrics calculation.
- struct METRIC_RESULTS_STRUCT

  This structure stores outputs results for metrics calculation.
- struct ESG_CURRENCY_RATE_INPUTS

  This structure stores specified inputs to setup ESG specified currency simulation interest rates.
- struct ESG_MODEL_INPUTS

  This structure stores inputs to setup ESG model interest rates.
- struct RATE_SHIFT_SETTING

  This structure stores inputs to setup ECON rate settings.
- struct FIRST_LOSS_INPUT
- struct FIRST_LOSS_RESULT
- struct CHINA_BOND_INFO
- struct PPC_STRUCT

Macros

- #define WSA_API_VERSION "4.0.1.0"

  WSA API Version.

Enumerations

- enum ENGINE_PREFERENCE { PICK_CHS_ENGINE_FOR_MAPPED_DEALS, PICK_SFW_ENGINE_FO-R_MAPPED_DEALS, PICK_CDONET_ENGINE_FOR_MAPPED_DEALS }
- enum ENGINE_TYPE {
  ENGINE_TYPE_ERROR = -99, UNKNOWN_ENGINE =-99, CHS_ENGINE = 0, SFW_ENGINE = 1,
  CDONET_ENGINE = 2 }
- enum INDEX_TYPE_EX {
  INDEX_TYPE_EX_BASE =MAX_INDEX_TYPES-1, GNMA_MORT, FNMA_MORT, FED_FUNDS,
  LARGE_CD, EURO_CD, COMM_PP, CORP_AAA,
  COMM_LOAN, NATCOFI, CD_6, AUTO_LOAN,
  MTA, XRATE1, XRATE2, XRATE3,
  XRATE4, XRATE5, XRATE6, XRATE7,
  XRATE8, INFLATION, SVR, BBR,
  COFI_6, NAT_MORT, LIBOR_2, LIBOR_1W,
  LIBOR_2W, LIBOR_4, BLR_12, BLR_12_60,
  BLR_60_PLUS, BDR_3, BDR_6, BDR_12,
  BDR_24, BDR_36, BDR_60, HPF_0_60,
  HPF_60_PLUS, MAX_INDEX_TYPES_EX }
- enum MISSING_EXCHANGE_RATES_HANDLING { MISSING_EXCHANGE_RATES_USE_ONE, MISSIN-G_EXCHANGE_RATES_TREAT_AS_ERROR, NUM_MISSING_EXCHANGE_RATES_TYPE }
- enum RESEC_EXCEPTIONS_HANDLING { RESEC_EXCEPTIONS_HANDLING_TREAT_AS_NONRESE-C, RESEC_EXCEPTIONS_HANDLING_TREAT_AS_ERROR, NUM_RESEC_EXCEPTIONS_HANDLING_-TYPE }
- enum POOL_PROPERTY_TYPES_EX {
  POOL_PROPERTY_TYPE_EX_BASE =POOL_PROPERTY_NOT_AVAILABLE, POOL_PROPERTY_MUL-TI_FAMILY_TWO, POOL_PROPERTY_MULTI_FAMILY_THREE, POOL_PROPERTY_MULTI_FAMILY_F-OUR,
  POOL_PROPERTY_MULTI_FAMILY_FIVEPLUS }
- enum POOL_DELINQ_STATES_EX {
  POOL_DELINQ_STATE_EX_BASE =POOL_DELINQ_STATES_SIZE-1, POOL_DELINQ_120P, POOL_D-ELINQ_150P, POOL_DELINQ_180P,
  POOL_DEFEASED, POOL_NON_PERFORMING_MATURED_BALLOON, POOL_DELINQ_0P, POOL_BA-NKRUPT,
  POOL_PAID_OFF, POOL_REPURCHASED, POOL_LIQUIDATED, POOL_CLOSED,
  POOL_DELINQ_STATES_EX_SIZE }

- enum POOL_DOCUM_TYPES_EX {
  POOL_DOCUM_TYPES_EX_BASE =POOL_DOCUM_UNKNOWN, POOL_DOCUM_FULL_ASSETS_PA-RTIAL_INCOME, POOL_DOCUM_FULL_INCOME_NO_ASSETS, POOL_DOCUM_NO_INCOME_PARTI-AL_ASSETS,
  POOL_DOCUM_NO_INCOME_STATED_ASSETS }
- enum PREPAY_DEFAULT_CALC_METHOD_TYPE {
  PREPAY_DEFAULT_BEFORE_SCHED_PRIN_PPYDEF, SCHED_PRIN_PREPAY_BEFORE_DEFAULT_-PPYDEF, DEFAULT_BEFORE_SCHED_PRIN_PREPAY_PPYDEF, DEFAULT_PREPAY_SCHED_PRIN_-PPYDEF,
  JAPANESE_PREPAY_DEFAULT_PPYDEF, NUM_PREPAY_DEFAULT_METHODS }
- enum DRAW_RATE_TYPE {
  DRAW_CURVE_SMM =0, DRAW_CURVE_CPR, DRAW_CURVE_SEASONED_SMM, DRAW_CURVE_S-EASONED_CPR,
  NUM_DRAW_RATE_TYPE }
- enum FLOW_MISC_INDENTIFIER {
  FLOW_MISC_FEE_TOTAL = 0, FLOW_MISC_FEE_AE, FLOW_MISC_FEE_DEF_AE, FLOW_MISC_FEE-_SCMF,
  FLOW_MISC_FEE_DEF_SCMF, FLOW_MISC_FEE_SUCMF, FLOW_MISC_FEE_DEF_SUCMF, FLOW_-MISC_FEE_ICMF,
  FLOW_MISC_FEE_DEF_ICMF, FLOW_MISC_FEE_ICMF2, FLOW_MISC_FEE_TAXES, FLOW_MISC_F-EE_TRUSTEE,
  FLOW_MISC_FEE_DEF_TRUSTEE }
- enum DEAL_ACCOUNT_TYPES { DEAL_RESERVE_ACCOUNT, DEAL_LIQFAC_ACCOUNT, DEAL_INS-URANCE_ACCOUNT, NUM_DEAL_ACCOUNT_TYPES }
- enum TEST_TYPE { TEST_IC, TEST_PV, TEST_UD }
- enum CDO_HAIRCUT_TYPE { HAIRCUT_1, HAIRCUT_2, HAIRCUT_UD }
- enum BUY_PRICE_OVERRIDE_TYPE {
  BUY_PRICE_OVERRIDE_NO, BUY_PRICE_OVERRIDE_MODEL, BUY_PRICE_OVERRIDE_MARKET, B-UY_PRICE_OVERRIDE_INPUT,
  NUM_BUY_PRICE_OVERRIDE_TYPE }
- enum CALL_DATE_OVERRIDE_TYPE {
  CALL_DATE_OVERRIDE_NO, CALL_DATE_OVERRIDE_MATURITY, CALL_DATE_OVERRIDE_NEXT_-CALLABLE, CALL_DATE_OVERRIDE_AUCTION,
  CALL_DATE_OVERRIDE_CLEAN_UP, CALL_DATE_OVERRIDE_INPUT, NUM_CALL_DATE_OVERRID-E_TYPE }
- enum CALL_PRICE_OVERRIDE_TYPE {
  CALL_PRICE_OVERRIDE_NO, CALL_PRICE_OVERRIDE_MODEL, CALL_PRICE_OVERRIDE_MARKE-T, CALL_PRICE_OVERRIDE_INPUT,
  NUM_CALL_PRICE_OVERRIDE_TYPE }
- enum REINV_TYPE { DEAL_REINV, GLOBAL_REINV, NUM_REINV_TYPE }
- enum REINV_TERM_SETTING_TYPE {
  TERM_NON_SEASONED_INPUT, TERM_SEASONED_INPUT, TERM_WAL_CURRENT, TERM_WAL_T-RIGGER,
  NUM_TERM_SETTING_TYPE }
- enum REINV_OVERRIDE_TYPE { REINV_OVERRIDE_ALWAYS, REINV_OVERRIDE_NEVER, REINV_O-VERRIDE_REINV_PER, NUM_REINV_OVERRIDE_TYPE }
- enum ASSET_SENIORITY {
  SENIORITY_BLANK, SENIORITY_SUB, SENIORITY_MEZZ, SENIORITY_JUNIOR,
  SENIORITY_SENIOR, SENIORITY_SR_SEC, SENIORITY_SR_SUB, SENIORITY_JR_MEZZ,
  SENIORITY_SR_MEZZ, SENIORITY_SR_UNSEC, SENIORITY_SOVEREIGN, SENIORITY_SECOND_LI-EN,
  SENIORITY_SUB_SEC, SENIORITY_SUB_UNSEC, NUM_ASSET_SENIORITY_TYPE }
- enum PAYMENT_FREQUENCY {
  PAY_FREQ_MONTHLY, PAY_FREQ_SANNUALLY, PAY_FREQ_QUARTERLY, PAY_FREQ_ANNUALLY,
  PAY_FREQ_BIMONTHLY, NUM_PAYMENT_FREQUENCY }

- enum MOODYS_CREDIT_MODEL_SETTINGS {
MOODYS_STANDARD_SETTINGS, MOODYS_DPLC_SETTINGS, MOODYS_CMM_SETTINGS, MOODYS_MPA_SETTINGS,
MOODYS_PA_SETTINGS, MOODYS_SEDF_SETTINGS, MOODYS_UK_MPA_SETTINGS, NUM_MOODYS_CREDIT_MODEL_SETTINGS }
- enum MPA_ANALYSIS_TYPE {
MPA_LOSS_SIMULATION = 0, MPA_MEDC_SINGLE_PATH, MPA_MEDC_SIMULATION, MPA_CUST_MEDC_SINGLE_PATH,
MPA_CUST_MEDC_SIMULATION, MPA_FIXED_ECO_SIMULATION, MPA_FIXED_CUST_SIMULATION, NUM_MPA_ANALYSIS_TYPE }
- enum MPA_MULTIPLIER_TYPE { MPA_MULTIPLIER_PREPAY = 0, MPA_MULTIPLIER_DEFAULT, MPA_MULTIPLIER_SEVERITY, NUM_MPA_MULTIPLIER_TYPE }
- enum MPA_ANALYSIS_PARAM { MPA_ANALYSIS_PARAM_PREPAY, MPA_ANALYSIS_PARAM_DEFAULT, MPA_ANALYSIS_PARAM_SEVERITY }
- enum MPA_ANALYSIS_PARAM_OFFSET { MPA_ANALYSIS_PARAM_OFFSET_LTV, MPA_ANALYSIS_PARAM_OFFSET_FICO }
- enum PA_ANALYSIS_TYPE {
PA_LOSS_SIMULATION = 0, PA_MEDC_SINGLE_PATH, PA_CUST_MEDC_SIMULATION, PA_CUST_MEDC_SINGLE_PATH,
PA_FIXED_ECO_SIMULATION, PA_FIXED_CUST_SIMULATION, NUM_PA_ANALYSIS_TYPE }
- enum NON_PERFORMING_STATUS {
NON_PERFORMING_DELINQUENT, **NON_PERFORMING_BANKRUPTED**, **NON_PERFORMING_REO**, **NON_PERFORMING_FORECLOSED**,
NON_PERFORMING_SIZE }
- enum TRANCHE_NULLIFICATION_TYPE { **NULL_TYPE_NO**, **NULL_TYPE_PAY**, **NULL_TYPE_FULL** }

  This structure stores information about tranche nullified type.
- enum CALL_OPTION_TYPE {
RUN_TO_MATURITY = 0, FORCED_EARLIEST_CALL, EARLIEST_CALL, CLEANUP_CALL,
FORCED_CLEANUP_CALL, STEPUP_CALL, FORCED_STEPUP_CALL, NUM_CALL_OPTION_TYPE }
- enum SIMULATION_TYPE { SIMULATION_MONTE_CARLO = 0, SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION, NUM_SIMULATION_TYPE }
- enum MONTE_CARLO_OPTIMIZATION { MC_OPTIMIZATION_NONE = 0, MC_OPTIMIZATION_PATHS, MC_OPTIMIZATION_TAIL_RUN, NUM_MC_OPTIMIZATION }
- enum MONTE_CARLO_DEFAULT_TYPE {
ASSET_DEFAULT_PROBABILITY = 0, MOODYS_RATING_DEFAULT_PROBABILITY, FITCH_RATING_DEFAULT_PROBABILITY, INDUSTRY_DEFAULT_PROBABIILTY,
NUM_MC_DEFAULT_TYPE }
- enum FFIEC_TEST_MODE { NO_BENCH_MODE = 0, INPUT_MODE, BENCH_MODE, AVG_MODE }

  Enum FFIEC test mode .
- enum MONTE_CARLO_CORRELATION_TYPE { MONTE_CARLO_CORRELATION_PORTFOLIO = 0, MONTE_CARLO_CORRELATION_INDUSTRY, NUM_MONTE_CARLO_CORRELATION_TYPE }

  Enum for monte carlo correlation types.
- enum DPD_DISTRIBUTION_TYPE { DPD_DISTRIBUTION_LOGNORMAL, DPD_DISTRIBUTION_INVERSE_NORMAL, DPD_DISTRIBUTION_USER_DEFINED, NUM_DPD_DISTRIBUTION_TYPE }

  Enum for Default Probability Distribution type.
- enum SERVICER_ADVANCES_BASE { SERVICER_ADVANCES_BASE_OFF = 0, SERVICER_ADVANCES_BASE_DELINQ = 1, SERVICER_ADVANCES_BASE_DEFAULT }

  Enum for the types of servicer advance rate.
- enum INSURANCE_CLAIM { INSURANCE_CLAIM_COVERAGE = 1, INSURANCE_CLAIM_ACCRUED }

  Enum for the types of insurance rate.
- enum INT_CAPITAL_CODE_OVERRIDE {
INT_CAPITAL_CODE_OVERRIDE_NONE = 0, INT_CAPITAL_CODE_OVERRIDE_REPAYMENT, INT_CAPITAL_CODE_OVERRIDE_MONTHLY, INT_CAPITAL_CODE_OVERRIDE_QUARTERLY,
INT_CAPITAL_CODE_OVERRIDE_SEMIANNUALLY, INT_CAPITAL_CODE_OVERRIDE_ANNUALLY }

  Enum for the types of interest capitalization code override.

- enum LIQUIDATION_PERIODICITY_TYPE {
LIQUIDATION_MONTHLY = 0, LIQUIDATION_BIMONTHLY = 8, LIQUIDATION_QUARTERLY = 1, LIQUIDATION_SEMI_ANNUALLY = 2,
LIQUIDATION_ANNUALLY = 3 }

  Enum for the types of liquidation periodicity.
- enum US_STATE {
US_STATE_NONE, **US_STATE_AL**, **US_STATE_AK**, **US_STATE_AZ**,
US_STATE_AR, **US_STATE_CA**, **US_STATE_CO**, **US_STATE_CT**,
US_STATE_DE, **US_STATE_FL**, **US_STATE_GA**, **US_STATE_HI**,
US_STATE_ID, **US_STATE_IL**, **US_STATE_IN**, **US_STATE_IA**,
US_STATE_KS, **US_STATE_KY**, **US_STATE_LA**, **US_STATE_ME**,
US_STATE_MD, **US_STATE_MA**, **US_STATE_MI**, **US_STATE_MN**,
US_STATE_MS, **US_STATE_MO**, **US_STATE_MT**, **US_STATE_NE**,
US_STATE_NV, **US_STATE_NH**, **US_STATE_NJ**, **US_STATE_NM**,
US_STATE_NY, **US_STATE_NC**, **US_STATE_ND**, **US_STATE_OH**,
US_STATE_OK, **US_STATE_OR**, **US_STATE_PA**, **US_STATE_RI**,
US_STATE_SC, **US_STATE_SD**, **US_STATE_TN**, **US_STATE_TX**,
US_STATE_UT, **US_STATE_VT**, **US_STATE_VA**, **US_STATE_WA**,
US_STATE_WV, **US_STATE_WI**, **US_STATE_WY**, **US_STATE_DC**,
US_STATE_PR }

  Enum for the types of state.
- enum LOAN_STATUS {
LOAN_STATUS_NONE, **LOAN_STATUS_CURRENT**, **LOAN_STATUS_UNKNOWN**, **LOAN_STATUS_PAIDOFF**,
LOAN_STATUS_DELINQUENT, **LOAN_STATUS_FORECLOSED**, **LOAN_STATUS_BANKRUPT**, **LOAN_STATUS_REO**,
LOAN_STATUS_DEFAULTED, **LOAN_STATUS_REPURCHASED**, **LOAN_STATUS_LIQUIDATED**, **LOAN_STATUS_CLOSED** }

  Enum for the types of loan status.
- enum WHOLE_LOAN_DEFAULT_METHOD_TYPE { **WL_NORMAL_DEFAULT_METHOD**, **DEFAULT_PATTERN_NONBINARY**, **DEFAULT_PATTERN_BINARY** }

  Enum for the whole loan default caculation method type.
- enum ESG_RATING_TYPE { **ESG_RATING_BBB**, **ESG_RATING_BB**, **ESG_RATING_B**, **ESG_RATING_CCC** }

  Enum for bank loan rating type.
- enum ESG_RATING_TERM {
ESG_TERM_1M = 1, **ESG_TERM_2M** = 2, **ESG_TERM_3M** = 3, **ESG_TERM_6M** = 6,
ESG_TERM_12M = 12 }

  Enum for bank loan term type.
- enum HECM_PAYMENT_PLAN {
PAYMENT_PLAN_TENURE, PAYMENT_PLAN_TERM, PAYMENT_PLAN_LOC, PAYMENT_PLAN_MOD_TENURE,
PAYMENT_PLAN_MOD_TERM }
- enum WHOLE_LOAN_AMORTIZATION_TYPE { ANN, LIN, BUL, BULINV }
- enum WHOLE_LOAN_OCCUPANCY_TYPE { WL_OCCUPANCY_UNK, WL_OCCUPANCY_OWN, WL_OCCUPANCY_SEC, WL_OCCUPANCY_INV }
- enum WHOLE_LOAN_COUPON_TYPE {
FIXED_COUPON, FLOATING_COUPON, FIXED_TO_FLOATING, FLOATING_TO_FIXED,
FIXED_STEP, FLOATING_STEP, FIXED_TO_FLOATING_STEP, FLOATING_TO_FLOATING_STEP }

  Enum for the types of coupon.
- enum UK_REGION {
IGBR, EAMI, EAST, LOND,
NEAS, NORW, NOIR, SCTL,
SOEA, SOWE, WALS, WEMI,
YOHU }

  Enum for the types of UK Region.

- enum MOODYS_RATING_TYPE {
MOODYS_RATING_Aaa, MOODYS_RATING_Aa1, MOODYS_RATING_Aa2, MOODYS_RATING_Aa3, MOODYS_RATING_A1, MOODYS_RATING_A2, MOODYS_RATING_A3, MOODYS_RATING_Baa1, MOODYS_RATING_Baa2, MOODYS_RATING_Baa3, MOODYS_RATING_Ba1, MOODYS_RATING_Ba2, MOODYS_RATING_Ba3, MOODYS_RATING_B1, MOODYS_RATING_B2, MOODYS_RATING_B3, MOODYS_RATING_Caa1, MOODYS_RATING_Caa2, MOODYS_RATING_Caa3, MOODYS_RATING_Ca, MOODYS_RATING_C, MOODYS_RATING_D, MOODYS_RATING_LD, MOODYS_RATING_WR, MOODYS_RATING_NULL }

  Enum for the types of MOODYS RATING.
- enum BANKLOAN_JUNIOR_SENIOR_TYPE {
NULL_JSTYPE, **JUNIOR_JSTYPE**, **MEZZ_JSTYPE**, **SENIOR_JSTYPE**, **SRSUB_JSTYPE**, **SUB_JSTYPE**, **SR_SEC_JSTYPE**, **SR_UNSEC_JSTYPE**, **SECOND_LIEN_JSTYPE** }

  Enum for the types of bank loan junior-senior type.
- enum WHOLE_LOAN_TYPE {
WL_RMBS, WL_ABS_AUTO, WL_ABS_STUDENT_LOAN, WL_ABS_CREDIT_CARD, WL_ABS_AUTO_LEASE, WL_CMBS, WL_CDO, WL_HECM, WL_REVERSE_MORTGAGE, WL_TYPE_OTHER }

  Enum for the types of loan.
- enum WHOLE_LOAN_ISSUER_TYPE {
WL_FNMA = 1, WL_FHLMC, WL_GNMA, WL_NA_SUBPRIME, WL_NA_PRIME, WL_ISSUER_OTHER }

  Enum for the issue type.
- enum MOODYS_SWAP_NOTIONAL_CODE {
SWAP_FIXED_LOAN_NOTIONAL, SWAP_FLOATING_LOAN_NOTIONAL, SWAP_ALL_ASSETS_NOTIO-NAL, SWAP_CUSTOM_NOTIONAL, SWAP_CAP_NOTIONAL, SWAP_FLOOR_NOTIONAL, **NUM_MOODYS_SWAP_NOTIONAL_CODES** }

  Enum for swap notional code.
- enum MOODYS_FEE_CAL_CODE {
FEES_CALC_NONE, FEES_TOTAL_POOL_CALC, FEES_TOTAL_BONDS_CALC, FEES_FIXED_AMOU-NT_CALC, FEES_CUSTOM_CALC, **NUM_FEES_CALC** }

  Enum for fee type calculate code.
- enum APPLY_SPREAD_TYPE { APPLY_SPREAD_TO_TSY, APPLY_SPREAD_TO_LIBOR, **NUM_APPL-Y_SPREAD_TO** }

  Enum for metrics spread type.
- enum OAS_CAL_MODE { ENABLE_NONE, OAS_ONLY, ENABLE_ALL, **NUM_OAS_MODE** }

  Enum for metrics calculation mode.
- enum METRIC_ANCHORS { OAS, MARKET_PRICE }

  Enum for the metrics input anchors type.
- enum RATING_AGENCY {
S_AND_P_CURRENT, **S_AND_P_ORIGINAL**, **MOODYS_CURRENT**, **MOODYS_ORIGINAL**, **FITCH_CURRENT**, **FITCH_ORIGINAL**, **DCR_CURRENT**, **DCR_ORIGINAL**, **DBRS_CURRENT**, **DBRS_ORIGINAL**, **MAX_RATINGAGENCY** }
- enum ESG_RATE_TYPE { SPOT_RATE, SPOT_SPREAD_RATE }
- enum MACROECONOMIC_FACTOR_TYPE {
REALGDPGROWTH, UNEMPRATE, FEDFUNDSRATE, CPIINFRATE, POPGROWTH, NUMHOUSEHOLDSGROWTH, RETAILSALESGROWTH, TOTNONFARMEMPGROWTH, PERSONALINCGROWTH, HOMEPRICEGROWTH, BAACORPYIELD, CREPXIDXGROWTH }
- enum SCENARIO_RATE_SHIFT_TYPE { **SCENARIO_DEAL**, **SCENARIO_YES**, **SCENARIO_NO** }
- enum FIRST_LOSS_THRESHOLD { FIRST_LOSS_THRESHOLD_INTEREST, FIRST_LOSS_THRESHOL-D_PRINCIPAL, FIRST_LOSS_THRESHOLD_EITHER }

- enum FIRST_LOSS_RUN_MODE {
FIRST_LOSS_RUN_MODE_PRINCIPAL_PAYMENT_RATE, FIRST_LOSS_RUN_MODE_MONTHLY_PURCHASE_RATE, FIRST_LOSS_RUN_MODE_PORTFOLIO_YIELD, FIRST_LOSS_RUN_MODE_LOSS_RATE,
FIRST_LOSS_RUN_MODE_DEFAULT, FIRST_LOSS_RUN_MODE_RECOVERY_RATE, FIRST_LOSS_RUN_MODE_LOSS_SEVERITY, FIRST_LOSS_RUN_MODE_PREPAY,
FIRST_LOSS_RUN_MODE_FORBEARANCE, FIRST_LOSS_RUN_MODE_DEFERMENT }
- enum SEASONING_TYPE { SEASONING_GLOBAL, SEASONING_NO, SEASONING_YES }
- enum PREPAY_DEFAULT_COMPOUNDING_METHOD { PREPAY_DEFAULT_COMPOUNDING_MONTHLY, PREPAY_DEFAULT_COMPOUNDING_PERIODICITY }

Functions

- void CHASAPI set_engine_preference (const ENGINE_PREFERENCE &engine)
- ENGINE_TYPE CHASAPI get_current_deal_engine (void ∗tid)
- int CHASAPI get_deal_account_avail (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗account_names[ ], DEAL_ACCOUNT_INFO account_info[ ], unsigned int account_size)
- int CHASAPI set_deal_account_default (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗account_name, BOOLYAN account_default)
- int CHASAPI set_service_reimburse_advint (void ∗tid, const char ∗reremic_deal_id_or_null, BOOLYAN reimburse_advint)
- int CHASAPI set_ppydef_only_on_paydate (void ∗tid, const char ∗reremic_deal_id_or_null, BOOLYAN only_on_paydate)
- long CHASAPI set_recovery_from (void ∗tid, short type, BOOLYAN set_sup_remic)
- long CHASAPI ignore_asset_nonpayment_term (void ∗tid, bool val)
- long CHASAPI set_forbearance_rates (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)
- long CHASAPI set_deferment_rates (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)
- long CHASAPI set_grace_rates (void ∗tid, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)
- long CHASAPI get_available_borrower_benefits (void ∗tid, const char ∗reremic_deal_id_or_null, BORROWER_BENEFIT_ELIGIBILITY benefit_list[ ], int size)
- long CHASAPI set_borrower_benefits_rate (void ∗tid, const char ∗reremic_deal_id_or_null, short index, short vector, double ∗pval)
- long CHASAPI set_insurance_coverage (void ∗tid, const char ∗issuer, INSURANCE_CLAIM type, short is_vector, double ∗pval)
- long CHASAPI enable_bond_insurance (void ∗tid, const char ∗bondid, BOOLYAN is_enabled)
- int CHASAPI set_cmm_custom_scenario (void ∗tid, CMM_FACTOR_TYPE cmm_factor_type, CMM_FACTOR factor, const double ∗value, int length)
- int CHASAPI generate_cmm_custom_result_output (void ∗tid, char ∗custom_scen_name)
- long CHASAPI set_credit_card_assump_ex (void ∗tid, const char ∗reremic_deal_id_or_null, CREDIT_CARD_ASSUMP_TYPE assump_type, short is_vector, double ∗pval, long loan_num)
- long CHASAPI set_moodys_credit_model_settings (void ∗tid, const MOODYS_CREDIT_MODEL_SETTINGS credit_model, BOOLYAN sets_up_only)
- long CHASAPI clear_moodys_credit_model_setttings (void ∗tid)
- long CHASAPI get_moodys_pool_group_id (void ∗tid, const char ∗reremic_deal_id_or_null, int group_number, char ∗group_id)
- int CHASAPI install_collat_assump_cb_ex1 (void ∗tid, COLLAT_ASSUMP_CB_EX1 collat_assump_cb_ex1)
- int CHASAPI get_deal_update_id (void ∗tid, char ∗const update_id, const int len)
- int CHASAPI get_moodys_id (const char ∗id, char ∗deal, int deal_length, char ∗bond, int bond_length, char ∗err_buffer, int err_length)
- long CHASAPI get_moodys_ssfa_calc (void ∗tid, const char ∗bondid, MOODYS_SSFA_CALC ∗ssfa_calc)
- long CHASAPI get_trustee_loan_id (void ∗tid, const char ∗reremic_deal_id_or_null, int loan_number, char ∗trustee_loan_id)

- long CHASAPI set_default_till_end (void ∗tid, BOOLYAN val, BOOLYAN set_sup_remic)
- long CHASAPI set_recover_at_maturity_call (void ∗tid, BOOLYAN is_enabled, BOOLYAN set_sup_remic)
- long CHASAPI set_liquidation_period (void ∗tid, const int period, long loan_num, BOOLYAN set_sup_remic)
- long CHASAPI set_liquidation_schedule (void ∗tid, short vector_length, double ∗pval, long loan_num, BOOLYAN set_sup_remic)
- int CHASAPI set_calculation_method (void ∗tid, PREPAY_DEFAULT_CALC_METHOD_TYPE method_index, BOOLYAN set_sup_remic)
- int CHASAPI get_calculation_method (void ∗tid, const char ∗reremic_deal_id_or_null)
- int CHASAPI set_realized_losses_at_liquidation (void ∗tid, BOOLYAN realized_at_liquidation, int months_prior_liquidation, BOOLYAN set_sup_remic)
- int CHASAPI set_liquidation_periodicity (void ∗tid, short liquidation_periodicity_type, BOOLYAN set_sup_remic)
- int CHASAPI get_moodys_cmm_scenarios (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗scenario_list[ ])
- int CHASAPI set_current_moodys_cmm_scenario (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗cmm_scenario)
- int CHASAPI get_current_moodys_cmm_scenario (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗cmm_scenario)
- int CHASAPI set_mpa_data_path (const char ∗path)
- int CHASAPI set_mpa_analysis_type (void ∗tid, MPA_ANALYSIS_TYPE type)
- int CHASAPI get_mpa_scenarios (void ∗tid, char ∗scenario_list[ ])
- int CHASAPI set_current_mpa_scenario (void ∗tid, int idx)
- int CHASAPI get_current_mpa_scenario (void ∗tid)
- int CHASAPI set_mpa_recovery_lag_by_state (void ∗tid, int judicial_lag, int non_judicial_lag)
- int CHASAPI set_mpa_recovery_lag (void ∗tid, short is_vector, int ∗pval, long loan_num)
- int CHASAPI set_mpa_multiplier (void ∗tid, MPA_MULTIPLIER_TYPE type, short is_vector, double ∗pval, long loan_num)
- int CHASAPI set_mpa_haircut (void ∗tid, short is_vector, double ∗pval, BOOLYAN seasoning)
- int CHASAPI set_mpa_simulation_length (void ∗tid, int length)
- int CHASAPI set_mpa_default_loan_data (void ∗tid, const char ∗loan_data_field, const char ∗value)
- int CHASAPI set_mpa_mid_course_adj (void ∗tid, BOOLYAN use)
- int CHASAPI set_mpa_custom_scenario (void ∗tid, const char ∗factor, const char ∗scope, const int ∗year, const int ∗quarter, const double ∗value, int length)
- int CHASAPI get_mpa_economy_date (void ∗tid, int ∗year, int ∗quarter)
- int CHASAPI set_mpa_simulation_path_num (void ∗tid, int number)
- int CHASAPI set_mpa_insurance_non_payment (void ∗tid, double probability)
- int CHASAPI set_mpa_optimization (void ∗tid, BOOLYAN toggle, double tail_percent, double opt_percent)
- int CHASAPI set_mpa_stress_range (void ∗tid, MPA_ANALYSIS_PARAM param_type, double floor, double cap)
- int CHASAPI set_mpa_delinquent_pd (void ∗tid, double deq_30days, double deq_60days)
- int CHASAPI set_mpa_offset (void ∗tid, MPA_ANALYSIS_PARAM_OFFSET type, int unit, double offset)
- int CHASAPI set_mpa_confidence_level (void ∗tid, double confidence_level)
- int CHASAPI get_group_issue_date (void ∗tid, int group_number, char ∗date)
- int CHASAPI get_moodys_pool_history_avail_YYYYMMs (void ∗tid, int groupNumber, int YYYYMMs[ ], int sizeOfYYYYMMs, int ∗numAvailable)
- int CHASAPI get_moodys_pool_history (void ∗tid, int groupNumber, MOODYS_POOL_HISTORY poolHistory[ ], int sizeOfHistoryArray, int YYYYMM)
- int CHASAPI get_moodys_bond_history_avail_YYYYMMs (void ∗tid, const char ∗bondId, int YYYYMMs[ ], int sizeOfYYYYMMs, int ∗numAvailable)
- int CHASAPI get_moodys_bond_history (void ∗tid, const char ∗bondId, MOODYS_BOND_HISTORY bondHistory[ ], int sizeOfHistoryArray, int YYYYMM)
- long CHASAPI view_moodys_student_loan_info (void ∗tid, short index, MOODYS_STUDENT_LOAN_INFO all_colls[ ], short length)
- long CHASAPI set_draw_rates (void ∗tid, short type, short is_vector, double ∗pval, long loan_num, BOOLYAN set_sup_remic)
- double ∗CHASAPI get_misc_flow (void ∗tid, int flow_identifier)

- int CHASAPI get_bond_authorized_integral_amount (void ∗tid, char ∗bondid, double ∗value)
- int CHASAPI get_global_currency (void ∗tid, char ∗currency_index)
- int CHASAPI get_currencies (void ∗tid, char ∗currencies[ ])
- int CHASAPI set_exchange_rate (void ∗tid, const char ∗currency, double val)
- int CHASAPI get_exchange_rate (void ∗tid, const char ∗currency, double ∗pval)
- long CHASAPI set_index_rate (void ∗tid, const char ∗currency, short ∗idx, short vector, double ∗pval)
- int CHASAPI load_MWSA_rates (void ∗tid, int yyyymmdd, BOOLYAN load_forward_curves)
- int CHASAPI load_MA_rates (void ∗tid, int yyyymmdd, const char ∗ma_scenario)
- long CHASAPI get_indices (void ∗tid, const char ∗currency, short ∗ps_rates)
- double ∗CHASAPI get_index_rate (void ∗tid, const char ∗currency, short ∗idx)
- int CHASAPI get_required_index_codes (void ∗tid, const char ∗currency, int ∗rate_codes, int size_of_array_-codes)
- void CHASAPI set_missing_exchange_rates_handling (MISSING_EXCHANGE_RATES_HANDLING handling)
- int CHASAPI set_default_before_amortization (void ∗tid, BOOLYAN def_bef_amort, BOOLYAN set_sup_-remic)
- int CHASAPI set_buy_price_override (void ∗tid, short override_type, double ∗price, int size)
- int CHASAPI set_call_date_override (void ∗tid, short override_type, char ∗override_date)
- int CHASAPI set_call_price_override (void ∗tid, short override_type, double ∗price, int size)
- int CHASAPI set_reinvestment_type (void ∗tid, short reinv_type)
- int CHASAPI get_cdo_test_info (void ∗tid, short ∗test_size, CDO_TEST_INFO ∗test_info)
- int CHASAPI get_cdo_test_flow (void ∗tid, TEST_TYPE test_type, const char ∗test_name, CDO_TEST_FL-OW ∗flow_test)
- double ∗CHASAPI get_haircut_flow (void ∗tid, CDO_HAIRCUT_TYPE haircut_type)
- int CHASAPI get_cdo_date_info (void ∗tid, const char ∗reremic_deal_id_or_null, CDO_DATE_INFO ∗date_-info)
- int CHASAPI set_global_reinvestment (void ∗tid, GLOBAL_REINVESTMENT_INFO reinv_info, short pool_-size, const GLOBAL_REINVESTMENT_ASSET_INFO ∗pool_info)
- int CHASAPI get_global_reinvestment (void ∗tid, GLOBAL_REINVESTMENT_INFO ∗reinv_info, short pool-_size, GLOBAL_REINVESTMENT_ASSET_INFO pool_info[ ])
- int CHASAPI set_pv_reinvest_override (void ∗tid, const char ∗bondid, short override_type)
- void CHASAPI set_resec_exceptions_handling (RESEC_EXCEPTIONS_HANDLING handling)
- int CHASAPI set_pa_data_path (const char ∗path)
- int CHASAPI set_pa_default_pool_data (void ∗tid, const char ∗paraName, const char ∗value)
- int CHASAPI get_pa_default_pool_data (void ∗tid, const char ∗paraName, char ∗value, int &len)
- int CHASAPI replace_pa_pool_data (void ∗tid, int poolID, const char ∗paraName, const char ∗value)
- int CHASAPI get_pa_scenarios (void ∗tid, char ∗scenario_list[ ])
- int CHASAPI set_current_pa_scenario (void ∗tid, int idx)
- int CHASAPI get_current_pa_scenario (void ∗tid)
- int CHASAPI set_pa_analysis_type (void ∗tid, PA_ANALYSIS_TYPE type)
- int CHASAPI set_pa_custom_scenario (void ∗tid, const char ∗factor, const int ∗year, const int ∗quarter, const double ∗value, int length)
- int CHASAPI set_pa_custom_scenario_ex (void ∗tid, const char ∗factor, const char ∗country, const char ∗region, const int ∗year, const int ∗quarter, const double ∗value, int length)
- int CHASAPI set_pa_simulation_path_num (void ∗tid, int number)
- int CHASAPI get_pa_economy_date (void ∗tid, int ∗year, int ∗quarter)
- double ∗CHASAPI get_pa_vector (void ∗tid, int group_number, PA_POOL_VECTOR_TYPE identifier)
- int CHASAPI set_pa_multiplier (void ∗tid, PA_MULTIPLIER_TYPE type, short is_vector, double ∗pval, long pool_num)
- int CHASAPI set_balloon_extension_assumptions (void ∗tid, const char ∗reremic_deal_id_or_null, int ∗months, double ∗rates, int length, int delay, long loan_num)
- int CHASAPI get_balloon_extension_assumptions (void ∗tid, const char ∗reremic_deal_id_or_null, int ∗months, double ∗rates, int length, int ∗delay, long loan_num)
- long CHASAPI set_call_option (void ∗tid, short type, BOOLYAN set_sup_remic)
- int CHASAPI get_custom_call_status (void ∗tid, const char ∗reremic_deal_id_or_null, BOOLYAN ∗status)

- int CHASAPI get_optional_redemption_date (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗date)
- int CHASAPI get_coupon_stepup_date (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗date)
- int CHASAPI get_deal_refinance_date (void ∗tid, int refinance_dates_array[ ], int num_dates)
- int CHASAPI set_simulation_engine (void ∗tid, short simulation_type)
- int CHASAPI set_monte_carlo_assumption (void ∗tid, const MONTE_CARLO_ASSUMPTION ∗basic_-assumption, const MONTE_CARLO_DEF_PPY_REC_ASSUMPTION ∗def_ppy_rec_assumption)
- int CHASAPI run_monte_carlo_simulation (void ∗tid)
- int CHASAPI get_monte_carlo_result (void ∗tid, const char ∗bondid, MONTE_CARLO_RESULT ∗result)
- double ∗CHASAPI get_bond_flow_sim (void ∗tid, short path, const char ∗bondid, int flow_identifier)
- double ∗CHASAPI get_collateral_flow_sim (void ∗tid, short path, int flow_identifier)
- int CHASAPI get_monte_carlo_global_issuers (void ∗tid, char ∗issuer_names[ ], short size)
- int CHASAPI set_monte_carlo_correlation (void ∗tid, MONTE_CARLO_CORRELATION_TYPE type, char ∗field1, char ∗field2, double correlation)
- int CHASAPI get_monte_carlo_correlation (void ∗tid, MONTE_CARLO_CORRELATION_TYPE type, char ∗field1, char ∗field2, double ∗correlation)
- int CHASAPI set_monte_carlo_default_time_and_recovery (void ∗tid, short num_path, short num_loan, short default_time, double recovery)
- int CHASAPI get_bond_cf_length (void ∗tid, short path, const char ∗bondid)
- int CHASAPI get_coll_cf_length (void ∗tid, short path)
- int CHASAPI get_edf_scenarios (void ∗tid, char ∗scenario_list[ ])
- int CHASAPI set_current_edf_scenario (void ∗tid, int idx)
- int CHASAPI get_current_edf_scenario (void ∗tid)
- int CHASAPI get_loan_edf (void ∗tid, const char ∗reremic_deal_id_or_null, long loan_num, double pd[ ], int length)
- int CHASAPI set_loan_edf (void ∗tid, const char ∗reremic_deal_id_or_null, long loan_num, double ∗pd, int length)
- int CHASAPI get_reinv_weighted_avg_pd (void ∗tid, long loan_num, double pd[ ])
- int CHASAPI get_reinv_recovery_rate (void ∗tid, long loan_num, double ∗recovery_rate)
- int CHASAPI set_edf_default_multiplier (void ∗tid, double multiplier)
- int CHASAPI get_trigger_avail_ex (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗trigger_names[ ], char ∗trigger_descs[ ], int ∗num_sub_triggers, int size)
- int CHASAPI get_master_trigger_info (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗trigger_-name, SBYTE ∗breached, char ∗sub_trigger_logic, char ∗sub_trigger_names[ ], char ∗sub_trigger_descs[ ], int size)
- int CHASAPI get_sub_trigger_info (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗sub_trigger_-name, char ∗sub_trigger_type, char ∗sub_trigger_operator, double ∗current_level, double ∗threshold, SBYTE ∗status, BOOLYAN ∗curable, SBYTE ∗override_type, int ∗override_date)
- int CHASAPI set_trigger_override_ex (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗sub_-trigger_name, SBYTE override_type, int override_date)
- void CHASAPI set_resec_underlying_level (int level)
- int CHASAPI run_default_probability_distribution (void ∗tid)
- int CHASAPI set_dpd_assumption (void ∗tid, const DPD_ASSUMPTION ∗assumption)
- int CHASAPI set_dpd_scenarios (void ∗tid, const DPD_SCENARIO ∗scenarios, short size_scenario)
- int CHASAPI set_dpd_current_default_timing (void ∗tid, const double ∗timing, short size_timing, BOOLYAN seasoning)
- int CHASAPI set_dpd_revolving_default_timing (void ∗tid, const double ∗timing, short size_timing, BOOLYAN seasoning)
- int CHASAPI set_dpd_threshold (void ∗tid, const char ∗rating, short year, double threshold)
- int CHASAPI set_dpd_el_pd_factors (void ∗tid, double el_factor, double pd_factor)
- int CHASAPI get_dpd_scenarios (void ∗tid, DPD_SCENARIO ∗scenarios, short size_scenarios)
- double ∗CHASAPI get_dpd_current_default_timing (void ∗tid)
- double ∗CHASAPI get_dpd_revolving_default_timing (void ∗tid)
- int CHASAPI get_dpd_results (void ∗tid, const char ∗bondid, DPD_RESULT ∗result)
- int CHASAPI get_dpd_threshold (void ∗tid, const char ∗rating, short year, double ∗threshold)
- int CHASAPI get_dpd_el_pd_factors (void ∗tid, double ∗el_factor, double ∗pd_factor)

- int CHASAPI set_loan_lgd (void ∗tid, const char ∗reremic_deal_id_or_null, long loan_num, double ∗lgd, int length)
- int ∗CHASAPI get_bond_cf_dates (void ∗tid, const char ∗bondid)
- int ∗CHASAPI get_coll_cf_dates (void ∗tid)
- int CHASAPI set_service_advances_rates_type (void ∗tid, short type)
- int CHASAPI set_service_advances_rates (void ∗tid, int group_number, short is_vector, double ∗pval)
- int CHASAPI get_bond_implied_loss (void ∗tid, const char ∗bondid, double ∗implied_loss)
- int CHASAPI set_int_capital_code_override (void ∗tid, short int_capital_code_override_type)
- int CHASAPI set_default_non_performing_loans (void ∗tid, BOOLYAN is_defaulted, short ∗non_perf_status, BOOLYAN set_sup_remic)
- int CHASAPI set_non_perf_recovery_lag (void ∗tid, short value, BOOLYAN set_sup_remic)
- int CHASAPI set_smooth_losses (void ∗tid, BOOLYAN status, BOOLYAN set_sup_remic)
- int CHASAPI get_pa_model_type (void ∗tid, char ∗pa_model_type, int pa_avail_vector[], int ∗avail_vector_-num)
- int ∗CHASAPI get_bond_payflag (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid)
- double ∗CHASAPI get_loan_flow (void ∗tid, int loan_number, const char ∗reremic_deal_id_or_null, int flow_-identifier)
- int CHASAPI get_loan_flow_ex (void ∗tid, int loan_number, MOODYS_LOAN_CASHFLOW ∗cf)
- int ∗CHASAPI get_loan_dates (void ∗tid, int loan_number)
- int CHASAPI get_loan_flow_size (void ∗tid, int loan_number)
- long CHASAPI get_deal_info_ex (void ∗tid, const char ∗reremic_deal_id_or_null, MOODYS_DEAL_INFO ∗deal_info)
- int CHASAPI set_whole_loan (void ∗tid, const WHOLE_LOAN_STRUCT ∗whole_loan, int length, int initial_-date)
- int CHASAPI set_distressed_property_recovery (void ∗tid, int loan_number, DISTRESSED_PROPERTY_R-ECOVERY ∗recovery_inputs)
- int ∗CHASAPI get_bond_rate_reset_dates (void ∗tid, const char ∗bondid)
- void CHASAPI set_sfw_dll_num (const int &num)
- void CHASAPI set_cdonet_dll_num (const int &num)
- void CHASAPI set_sfw_unload_flag (bool unload_dll)
- void CHASAPI set_cdonet_unload_flag (bool unload_dll)
- int CHASAPI get_bond_currency (void ∗tid, const char ∗bondid, char ∗currency)
- int CHASAPI get_bond_step_up_coupon (void ∗tid, const char ∗bondid, BOND_STEP_UP_COUPON all_-set_up_coupons[], int array_size, int ∗num_available)
- int CHASAPI get_deal_hedge (void ∗tid, const char ∗reremic_deal_id_or_null, MOODYS_HEDGE_STRUCT hedge_info[], int size, int ∗num_hedges)
- int CHASAPI set_deal_hedge_override (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗hedge_-id, MOODYS_HEDGE_OVERRIDE hedge_override_info)
- int CHASAPI get_deal_fee (void ∗tid, const char ∗reremic_deal_id_or_null, MOODYS_FEE_STRUCT fee_-info[], int size, int ∗num_fees)
- double ∗CHASAPI get_deal_fee_flow (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗fee_name)
- int CHASAPI set_deal_fee_override (void ∗tid, const char ∗reremic_deal_id_or_null, int fee_id, short fee_-type, double override_value)
- int CHASAPI set_metrics_input_ex (void ∗tid, METRIC_INPUT_STRUCT_EX ∗metric_inputs)
- int CHASAPI set_bankloan_call_adj_param (void ∗tid, const BANKLOAN_CALL_ADJ_PARAM ∗bankloan_-adj, int length)
- int CHASAPI get_bond_market_risk_metrics (void ∗tid, const char ∗bondid, METRIC_INPUT_STRUCT ∗metric_inputs, METRIC_RESULTS_STRUCT ∗metric_results)
- int CHASAPI get_loan_market_risk_metrics (void ∗tid, int LoanID, METRIC_INPUT_STRUCT ∗metric_-inputs, METRIC_RESULTS_STRUCT ∗metric_results)
- int CHASAPI get_bond_market_risk_metrics_ex (void ∗tid, char ∗bondid, METRIC_ANCHORS anchor_type, double anchor_value, APPLY_SPREAD_TYPE apply_to, METRIC_RESULTS_STRUCT_EX ∗results_ex)
- int CHASAPI get_loan_market_risk_metrics_ex (void ∗tid, int LoanID, METRIC_ANCHORS anchor_type, double anchor_value, APPLY_SPREAD_TYPE apply_to, METRIC_RESULTS_STRUCT_EX ∗results_ex)
- int CHASAPI set_mpa_thread_count (void ∗tid, int number)
- int CHASAPI set_pa_thread_count (void ∗tid, int number)

- long CHASAPI enable_sfw_delinq_projection (void ∗tid, BOOLYAN is_enabled)
- int CHASAPI get_bond_rating_by_tranche (void ∗tid, const char ∗bondid, RATING_AGENCY agency, char ∗rating)
- int CHASAPI set_indiv_recovery_nonperf (void ∗tid, BOOLYAN use_indiv_recovery_nonperf)
- int CHASAPI enable_reinv_loan (void ∗tid, BOOLYAN populate_reinv_loan)
- int CHASAPI set_mpa_loan_cashflow (void ∗tid, BOOLYAN enable_loan_cf)
- int CHASAPI set_loan_schedule (void ∗tid, long loan_number, WHOLE_LOAN_SINK_FUND ∗sink_fund_-info)
- int CHASAPI set_whole_loan_default_method (void ∗tid, WHOLE_LOAN_DEFAULT_METHOD_TYPE type-_index)
- int CHASAPI set_whole_loan_cumulative_rate (void ∗tid, double val, long loan_num)
- int CHASAPI set_whole_loan_default_timing (void ∗tid, short vector_length, double ∗pval)
- void ∗CHASAPI obtain_collat_iterator_ex (void ∗tid, const char ∗reremic_deal_id_or_null)
- MOODYS_POOL_INFO ∗CHASAPI get_next_collat_ex (void ∗tid, void ∗collat_iterator)
- int CHASAPI get_bond_info_by_tranche_ex (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, MOODYS_BOND_INFO ∗bond_info)
- int CHASAPI get_bond_info_by_index_ex (void ∗tid, const char ∗reremic_deal_id_or_null, int index, MOOD-YS_BOND_INFO ∗bond_info)
- int CHASAPI set_index_rate_ex (void ∗tid, const char ∗currency, short ∗idx, int num_paths, short rate_size, double ∗∗idx_val)
- int CHASAPI set_spot_spread (void ∗tid, const char ∗currency, ESG_RATING_TYPE rating_type, ESG_RA-TING_TERM term_type, int num_paths, short rate_size, double ∗∗idx_val)
- int CHASAPI run_FFIEC_test (void ∗tid, int prepay_type, double ∗prepay_rates)
- int CHASAPI get_bond_FFIEC_results (void ∗tid, const char ∗bondid, FFIEC_INPUT_PARAMS ∗FFIEC_-inputs, FFIEC_RESULTS FFIEC_results[ ])
- int CHASAPI set_non_call_end (void ∗tid, int non_call_end_date)
- int CHASAPI get_bond_next_reset_date (void ∗tid, const char ∗bondid, int ∗next_reset_date)
- int CHASAPI set_up_ESG_model_interest_rates (ESG_MODEL_INPUTS ∗esg_inputs, ESG_CURRENCY-_RATE_INPUTS esg_currency_inputs[ ], int esg_currency_inputs_size)
- int CHASAPI set_global_rates (const char ∗currency, short rate_size, short ∗rate_types, double ∗rate_values)
- int CHASAPI set_FRA (void ∗tid, const char ∗currency, const char ∗rate_type, short start_month, short end-_month, double rate_value)
- int CHASAPI generate_forward_interest_rates (void ∗tid)
- double ∗CHASAPI get_forward_interest_rates (void ∗tid, const char ∗currency, short ∗rate_type)
- int CHASAPI get_asset_type_list (char ∗asset_type_list[ ], char ∗err_buffer, int err_length)
- int CHASAPI price_loan (void ∗tid, int loan_number, PRICING_ANCHORS anchorType, double anchorValue, PRICING_RESULTS ∗results)
- int CHASAPI price_loan_ex (void ∗tid, int loan_number, LOAN_PRICING_INPUT pricing_param_input, PRI-CING_RESULTS ∗results)
- int CHASAPI get_loan_next_reset_date (void ∗tid, int loan_number, int ∗next_reset_date)
- int CHASAPI set_cmbs_loan_extension_assumption (void ∗tid, BOOLYAN use_default, BOOLYAN apply_-flag, BOOLYAN non_perf_loan, int maturity_cutoff, int extend_years, double edf_threshold)
- int CHASAPI set_macroeconomic_factor_ex (void ∗tid, const char ∗country, short ∗factor_type, int num_-paths, short val_size, double ∗∗factor_val)
- int CHASAPI set_rate_shift_setting (void ∗tid, RATE_SHIFT_SETTING rate_shift)
- int CHASAPI enable_periodic_coupon_rate_projection (void ∗tid, BOOLYAN flag_periodic_rate)
- double ∗CHASAPI load_MWSA_rate (const char ∗file_path, int yyyymmdd, const char ∗currency, short ∗idx)
- double ∗CHASAPI load_MA_rate (const char ∗file_path, int yyyymmdd, const char ∗ma_scenario, const char ∗currency, short ∗idx)
- int CHASAPI get_MA_rate_shifts_scenarios (const char ∗file_path, int yyyymmdd, char ∗scenario_list[ ])
- int CHASAPI calculate_bond_first_loss (void ∗tid, const char ∗bondid, FIRST_LOSS_INPUT first_loss_input, FIRST_LOSS_RESULT ∗first_loss_result)
- int CHASAPI get_china_bond_info_by_tranche (void ∗tid, const char ∗reremic_deal_id_or_null, const char ∗bondid, CHINA_BOND_INFO ∗bond_info)
- int CHASAPI remove_simulation_cashflow_populated_limit (void ∗tid, BOOLYAN flag)

- int CHASAPI get_prospectus_prepayment_curves (void ∗tid, const char ∗reremic_deal_id_or_null, PPC_STRUCT all_PPCs[ ], int size, int ∗num_curves)
- int CHASAPI set_prospectus_prepayment_curves (void ∗tid, short PPC_index, int loan_num, BOOLYAN set_sup_remic)
- int CHASAPI adjust_PA_vectors (void ∗tid, bool enable)
- int CHASAPI use_spot_values_for_initial_period (void ∗tid, bool enable)
- int CHASAPI enable_default_on_snapshot_date (void ∗tid, BOOLYAN flag_snapshot)
- int CHASAPI set_prepay_default_compounding_method (void ∗tid, PREPAY_DEFAULT_COMPOUNDING_METHOD prepay_default_compound)
- int CHASAPI get_deal_account_flow (void ∗tid, const char ∗reremic_deal_id_or_null, char ∗account_name, MOODYS_ACCOUNT_CASHFLOW ∗cf)
- int CHASAPI get_bond_total_loss (void ∗tid, const char ∗bondid, double ∗total_loss)
- int CHASAPI get_first_principal_pay_month (void ∗tid, const char ∗bondid, char ∗first_prin_pay_month)
- int CHASAPI get_last_principal_pay_month (void ∗tid, const char ∗bondid, char ∗last_prin_pay_month)

22.3.1 Detailed Description

Version

4.0.1.0

Date

2011-2019

22.3.2 Enumeration Type Documentation

22.3.2.1 enum APPLY_SPREAD_TYPE

See Also

- get_bond_market_risk_metrics()
- get_bond_market_risk_metrics_ex()

Enumerator

APPLY_SPREAD_TO_TSY	Apply spread to Treasury 3 months Curves.
APPLY_SPREAD_TO_LIBOR	Apply spread to Libor 1 month Curves.

22.3.2.2 enum ASSET_SENIORITY

Seniority of the asset

New feature Subject to change

See Also

GLOBAL_REINVESTMENT_ASSET_INFO

Enumerator

SENIORITY_BLANK	Blank.
SENIORITY_SUB	Sub.
SENIORITY_MEZZ	Mezzanine.
SENIORITY_JUNIOR	Junior.

SENIORITY_SENIOR	Senior.
SENIORITY_SR_SEC	Senior secured.
SENIORITY_SR_SUB	Senior subordinated.
SENIORITY_JR_MEZZ	Junior Mezzanine.
SENIORITY_SR_MEZZ	Senior Mezzanine.
SENIORITY_SR_UNSEC	Senior unsecured.
SENIORITY_SOVEREIGN	Sovereign.
SENIORITY_SECOND_LIEN	Second lien loan.
SENIORITY_SUB_SEC	Subordinated secured.
SENIORITY_SUB_UNSEC	Subordinated unsecured.
NUM_ASSET_SENIORITY_TYPE	Number of asset seniority types.

22.3.2.3 enum BANKLOAN_JUNIOR_SENIOR_TYPE

See Also

set_whole_loan()

New feature Subject to change

22.3.2.4 enum BUY_PRICE_OVERRIDE_TYPE

Buy price override types.

New feature Subject to change

See Also

set_buy_price_override()

Enumerator

BUY_PRICE_OVERRIDE_NO	No: use the deal level assumptions.
BUY_PRICE_OVERRIDE_MODEL	MODEL: use 100.
BUY_PRICE_OVERRIDE_MARKET	MARKET: use weighted average market price of the asset pool.
BUY_PRICE_OVERRIDE_INPUT	INPUT: use the price vector provided by user.
NUM_BUY_PRICE_OVERRIDE_TYPE	Number of buy price override types.

22.3.2.5 enum CALL_DATE_OVERRIDE_TYPE

Call date override types.

New feature Subject to change

See Also

set_call_date_override()

Enumerator

CALL_DATE_OVERRIDE_NO NO: use the deal level assumptions.

CALL_DATE_OVERRIDE_MATURITY MATURITY: use deal maturity date as call date.

CALL_DATE_OVERRIDE_NEXT_CALLABLE NEXT_CALLABLE: use next callable date as call date.

CALL_DATE_OVERRIDE_AUCTION AUCTION: use auction call date as call date.

CALL_DATE_OVERRIDE_CLEAN_UP CLEAN_UP: use clean up call date as call date.

CALL_DATE_OVERRIDE_INPUT INPUT: use user input Date as call date.

NUM_CALL_DATE_OVERRIDE_TYPE Number of buy price override types.

22.3.2.6 enum CALL_OPTION_TYPE

Call option Type

New feature Subject to change

See Also

set_call_option

Enumerator

RUN_TO_MATURITY Run the deal to maturity.

FORCED_EARLIEST_CALL Call the deal if either the clean-up call or the step-up call conditions are met (even if the collateral available is not enough to cover the senior tranches)

EARLIEST_CALL Call the deal if either the clean-up call or the step-up call conditions are met and the collateral available is enough to cover the senior tranches.

CLEANUP_CALL Call the deal if the clean-up call conditions are met and the collateral available is enough to cover the senior tranches.

FORCED_CLEANUP_CALL Call the deal if the clean-up call conditions are met (even if the collateral available is not enough to cover the senior tranches)

STEPUP_CALL Call the deal if the step-up call conditions are met and the collateral available is enough to cover the senior tranches.

FORCED_STEPUP_CALL Call the deal if the step-up call conditions are met (even if the collateral available is not enough to cover the senior tranches)

NUM_CALL_OPTION_TYPE Number of call option types.

22.3.2.7 enum CALL_PRICE_OVERRIDE_TYPE

Call price override types.

New feature Subject to change

See Also

set_call_price_override()

Enumerator

CALL_PRICE_OVERRIDE_NO No: use the deal level assumptions.

CALL_PRICE_OVERRIDE_MODEL MODEL: use 100.

CALL_PRICE_OVERRIDE_MARKET MARKET: use weighted average market price of the asset pool.

CALL_PRICE_OVERRIDE_INPUT INPUT: use the price vector provided by user.

NUM_CALL_PRICE_OVERRIDE_TYPE Number of call price override types.

22.3.2.8 enum CDO_HAIRCUT_TYPE

CDO haircut type.

New feature Subject to change

Enumerator

HAIRCUT_1 Haircut 1.

HAIRCUT_2 Haircut 2.

HAIRCUT_UD User defined haircut.

22.3.2.9 enum DEAL_ACCOUNT_TYPES

Deal account types.

See Also

DEAL_ACCOUNT_INFO

Enumerator

DEAL_RESERVE_ACCOUNT Reserve Account.

DEAL_LIQFAC_ACCOUNT Liquidation Facility.

DEAL_INSURANCE_ACCOUNT Insurance Account.

NUM_DEAL_ACCOUNT_TYPES Number of account types.

22.3.2.10 enum DPD_DISTRIBUTION_TYPE

New feature Subject to change

Enumerator

DPD_DISTRIBUTION_LOGNORMAL Lognormal distribution.

DPD_DISTRIBUTION_INVERSE_NORMAL Inverse normal distribution.

DPD_DISTRIBUTION_USER_DEFINED User defined distribution.

NUM_DPD_DISTRIBUTION_TYPE Number of supported distribution type.

22.3.2.11 enum DRAW_RATE_TYPE

Type of draw rate

New feature Subject to change

See Also

set_draw_rates()

Enumerator

DRAW_CURVE_SMM Draw rate expressed in SMM.

DRAW_CURVE_CPR Draw rate expressed in CPR.

DRAW_CURVE_SEASONED_SMM Draw rate expressed in SMM with seasoned.

DRAW_CURVE_SEASONED_CPR Draw rate expressed in CPR with seasoned.

NUM_DRAW_RATE_TYPE Size of DRAW_RATE_TYPE.

22.3.2.12 enum ENGINE_PREFERENCE

This enumeration can be used to indicate preferred engine for deals supported by both CHS and SFW. By default, SFW is the preferred engine.

See Also

set_engine_preference()

Enumerator

PICK_CHS_ENGINE_FOR_MAPPED_DEALS Choose CHS as preferred engine to run deals.

PICK_SFW_ENGINE_FOR_MAPPED_DEALS Choose SFW as preferred engine to run deals.

PICK_CDONET_ENGINE_FOR_MAPPED_DEALS Choose CDONET as preferred engine to run deals.

22.3.2.13 enum ENGINE_TYPE

This enumeration can be used to indicate the engine opening current deal.

See Also

get_current_deal_engine()

Enumerator

ENGINE_TYPE_ERROR Obsolete, use UNKNOWN_ENGINE.

UNKNOWN_ENGINE Unknown engine (no deal open yet)

CHS_ENGINE Current deal opened by CHS engine.

SFW_ENGINE Current deal opened by SFW engine.

CDONET_ENGINE Current deal opened by CDONET engine.

22.3.2.14 enum ESG_RATE_TYPE

See Also

ESG_MODEL_INPUTS

Enumerator

SPOT_RATE Only spot rate.

SPOT_SPREAD_RATE Spot rate and credit model spread rate.

22.3.2.15 enum ESG_RATING_TERM

New feature Subject to change

22.3.2.16 enum ESG_RATING_TYPE

New feature Subject to change

22.3.2.17 enum FFIEC_TEST_MODE

See Also

get_bond_FFIEC_results()

Enumerator

NO_BENCH_MODE FFIEC test results base on none bench mode.

INPUT_MODE FFIEC test results base on input tsy yield mode.

BENCH_MODE FFIEC test results base on bench mode.

AVG_MODE FFIEC test results base on average life mode.

22.3.2.18 enum FIRST_LOSS_RUN_MODE

Enumerator

FIRST_LOSS_RUN_MODE_PRINCIPAL_PAYMENT_RATE Principal payment rate.

FIRST_LOSS_RUN_MODE_MONTHLY_PURCHASE_RATE Monthly purchase rate.

FIRST_LOSS_RUN_MODE_PORTFOLIO_YIELD Portfolio yield.

FIRST_LOSS_RUN_MODE_LOSS_RATE Loss rate.

FIRST_LOSS_RUN_MODE_DEFAULT Default rate.

FIRST_LOSS_RUN_MODE_RECOVERY_RATE Recovery rate.

FIRST_LOSS_RUN_MODE_LOSS_SEVERITY Loss severity rate.

FIRST_LOSS_RUN_MODE_PREPAY Prepayment rate.

FIRST_LOSS_RUN_MODE_FORBEARANCE Forbearance rate.

FIRST_LOSS_RUN_MODE_DEFERMENT Deferment rate.

22.3.2.19 enum FIRST_LOSS_THRESHOLD

Enumerator

FIRST_LOSS_THRESHOLD_INTEREST Interest.

FIRST_LOSS_THRESHOLD_PRINCIPAL Principal.

FIRST_LOSS_THRESHOLD_EITHER either interest or principal

22.3.2.20 enum FLOW_MISC_INDENTIFIER

Deal level miscellaneous cashflow identifiers

New feature Subject to change

See Also

get_misc_flow()

Enumerator

FLOW_MISC_FEE_TOTAL Total fee (SFW and CDOnet)

FLOW_MISC_FEE_AE Administrative Fee (CDOnet only)

FLOW_MISC_FEE_DEF_AE Deferred Administrative Fee (CDOnet only)

FLOW_MISC_FEE_SCMF Senior Collateral Management Fees (CDOnet only)

FLOW_MISC_FEE_DEF_SCMF Deferred Senior Collateral Management Fees (CDOnet only)

FLOW_MISC_FEE_SUCMF Subordinate Collateral Management Fees (CDOnet only)

FLOW_MISC_FEE_DEF_SUCMF Deferred Subordinate Collateral Management Fees (CDOnet only)

FLOW_MISC_FEE_ICMF Incentive Collateral Management Fee (CDOnet only)

FLOW_MISC_FEE_DEF_ICMF Deferred Incentive Collateral Management Fee (CDOnet only)

FLOW_MISC_FEE_ICMF2 Incentive Collateral Management Fee 2 (CDOnet only)

FLOW_MISC_FEE_TAXES Taxes Fee (CDOnet only)

FLOW_MISC_FEE_TRUSTEE Trustee Fee (CDOnet only)

FLOW_MISC_FEE_DEF_TRUSTEE Deferred Trustee Fee (CDOnet only)

22.3.2.21 enum HECM_PAYMENT_PLAN

Enumerator

PAYMENT_PLAN_TENURE Equal monthly payments as long as at least one borrower lives and continues to occupy the property as a principal residence.

PAYMENT_PLAN_TERM Equal monthly payments for a fixed period of months selected.

PAYMENT_PLAN_LOC Unscheduled payments or in installments, at times and in an amount of your choosing until the line of credit is exhausted.

PAYMENT_PLAN_MOD_TENURE Combination of line of credit and scheduled monthly payments for as long as you remain in the home.

PAYMENT_PLAN_MOD_TERM Combination of line of credit plus monthly payments for a fixed period of months selected by the borrower.

22.3.2.22 enum INDEX_TYPE_EX

INDEX_TYPE_EX is the extension to INDEX_TYPE (In indextypes.h).

To get required index rate type codes (including extended ones), use get_required_rate_codes().

Enumerator

INDEX_TYPE_EX_BASE	Place holder.
GNMA_MORT	Ginnie Mae Mortgage Rate.
FNMA_MORT	Fannie Mae Mortgage Rate.
FED_FUNDS	Federal funds rate.
LARGE_CD	Large CD rate (Certificates of Deposit)
EURO_CD	Euro CD rate (Certificates of Deposit)
COMM_PP	Commercial Paper Rates.
CORP_AAA	Yield on AAA Corporate bond rate.
COMM_LOAN	Commercial loan rate.
NATCOFI	COFI rate (Cost of Funds Index)
CD_6	6 months CD rate (Certificates of Deposit)
AUTO_LOAN	Auto loan rate.
MTA	Monthly Treasury Average, aka Credit Card Index.
XRATE1	Extra Rate 1.
XRATE2	Extra Rate 2.
XRATE3	Extra Rate 3.
XRATE4	Extra Rate 4.
XRATE5	Extra Rate 5.
XRATE6	Extra Rate 6.
XRATE7	Extra Rate 7.
XRATE8	Extra Rate 8.
INFLATION	Inflation rate.
SVR	Standard Variable Rate.
BBR	Bank base rate.
COFI_6	6 months COFI rate (Cost of Funds Index)
NAT_MORT	National Mortgage rate.
LIBOR_2	2 months Libor rate
LIBOR_1W	1 week Libor rate
LIBOR_2W	2 week Libor rate
LIBOR_4	4 months Libor rate
BLR_12	China 1 year lending rate, SFW only.
BLR_12_60	China 1 - 5 year lending rate, SFW only.
BLR_60_PLUS	China 5+ year lending rate, SFW only.
BDR_3	China 3 months deposit rate, SFW only.
BDR_6	China 6 months deposit rate, SFW only.
BDR_12	China 1 year deposit rate, SFW only.
BDR_24	China 2 year deposit rate, SFW only.
BDR_36	China 3 year deposit rate, SFW only.
BDR_60	China 5 year deposit rate, SFW only.
HPF_0_60	China 0 - 5 year home provident fund rate, SFW only.
HPF_60_PLUS	China 5+ year home provident fund rate, SFW only.
MAX_INDEX_TYPES_EX	Number of index type(include both basic and extra index type)

22.3.2.23 enum INSURANCE_CLAIM

See Also

set_insurance_coverage()

New feature Subject to change

Enumerator

INSURANCE_CLAIM_COVERAGE Indicate vector/constant is coverage claim.

INSURANCE_CLAIM_ACCRUED Indicate vector/constant is accrued claim.

22.3.2.24 enum INT_CAPITAL_CODE_OVERRIDE

See Also

set_int_capital_code_override()

New feature Subject to change

Enumerator

INT_CAPITAL_CODE_OVERRIDE_NONE Turn off capitalization code override.

INT_CAPITAL_CODE_OVERRIDE_REPAYMENT Change the capitalization code to repayment.

INT_CAPITAL_CODE_OVERRIDE_MONTHLY Change the capitalization code to monthly.

INT_CAPITAL_CODE_OVERRIDE_QUARTERLY Change the capitalization code to quarterly.

INT_CAPITAL_CODE_OVERRIDE_SEMIANNUALLY Change the capitalization code to semi-annually.

INT_CAPITAL_CODE_OVERRIDE_ANNUALLY Change the capitalization code to annually.

22.3.2.25 enum LIQUIDATION_PERIODICITY_TYPE

See Also

set_liquidation_periodicity()

New feature Subject to change

Enumerator

LIQUIDATION_MONTHLY Liquidation periodicity is monthly.

LIQUIDATION_BIMONTHLY Liquidation periodicity is bimonthly.

LIQUIDATION_QUARTERLY Liquidation periodicity is quarterly.

LIQUIDATION_SEMI_ANNUALLY Liquidation periodicity is semi-annually.

LIQUIDATION_ANNUALLY Liquidation periodicity is annually.

22.3.2.26 enum LOAN_STATUS

See Also

set_whole_loan()

New feature Subject to change

22.3.2.27 enum MACROECONOMIC_FACTOR_TYPE

See Also

set_macroeconomic_factor_ex

Enumerator

REALGDPGROWTH	NIPA:Gross Domestic Product (growth).
UNEMPRATE	U3 - Unemployment Level.
FEDFUNDSRATE	Federal funds rate.
CPIINFRATE	Consumer Price Index (inflation).
POPGROWTH	Population growth.
NUMHOUSEHOLDSGROWTH	Number of households (growth)
RETAILSALESGROWTH	Retail sales (growth)
TOTNONFARMEMPGROWTH	Nonfarm employment (growth)
PERSONALINCGROWTH	Personal income (growth)
HOMEPRICEGROWTH	HPI (growth)
BAACORPYIELD	Moody's Intermediate-Term Bond Yield Average: Corporate - Rated Baa.
CREPXIDXGROWTH	Moody's/RCA Commercial Property Price Index - All property.

22.3.2.28 enum METRIC_ANCHORS

See Also

- get_bond_market_risk_metrics_ex()

Enumerator

OAS	Input type is OAS (in basis point).
MARKET_PRICE	Input type is market price.

22.3.2.29 enum MISSING_EXCHANGE_RATES_HANDLING

Missing exchange rates handling type

New feature Subject to change

See Also

set_missing_exchange_rates_handling

Enumerator

MISSING_EXCHANGE_RATES_USE_ONE	Use 1.0 as the exchange rate.
MISSING_EXCHANGE_RATES_TREAT_AS_ERROR	Report as an error.
NUM_MISSING_EXCHANGE_RATES_TYPE	Enumeration number of the type.

22.3.2.30 enum MONTE_CARLO_CORRELATION_TYPE

New feature Subject to change

Enumerator

MONTE_CARLO_CORRELATION_PORTFOLIO	Correlation portfolio.
MONTE_CARLO_CORRELATION_INDUSTRY	Correlation industry.
NUM_MONTE_CARLO_CORRELATION_TYPE	Number of monte carlo correlation type.

22.3.2.31 enum MONTE_CARLO_DEFAULT_TYPE

Monte carlo default types

New feature Subject to change

See Also

MONTE_CARLO_DEF_PPY_REC_ASSUMPTION

Enumerator

ASSET_DEFAULT_PROBABILITY Use loan level default probability set by set_loan_edf()

MOODYS_RATING_DEFAULT_PROBABILITY Use loan level default probability base on Moodys rating.

FITCH_RATING_DEFAULT_PROBABILITY Use loan level default probability base on Fetch rating.

INDUSTRY_DEFAULT_PROBABIILTY Reserve for future use.

NUM_MC_DEFAULT_TYPE Number of default probability source.

22.3.2.32 enum MONTE_CARLO_OPTIMIZATION

Monte carlo optimization types

New feature Subject to change

See Also

MONTE_CARLO_ASSUMPTION

Enumerator

MC_OPTIMIZATION_NONE No optimization.

MC_OPTIMIZATION_PATHS Percentage of Paths.

MC_OPTIMIZATION_TAIL_RUN Percentage of Tail Run.

NUM_MC_OPTIMIZATION Number of optimization.

22.3.2.33 enum MOODYS_CREDIT_MODEL_SETTINGS

Credit model settings

See Also

set_moodys_credit_model_settings()

Enumerator

MOODYS_STANDARD_SETTINGS Standard initial settings.

MOODYS_DPLC_SETTINGS DPLC model settings. Note that these settings (months to liquidation, delinquency rates, default non-performing loans flag, etc.) vary from asset class to asset class.

MOODYS_CMM_SETTINGS CMM model settings.

MOODYS_MPA_SETTINGS MPA model settings.

MOODYS_PA_SETTINGS PA model settings.

MOODYS_SEDF_SETTINGS SEDF model settings.

MOODYS_UK_MPA_SETTINGS MPA model settings for UK.

NUM_MOODYS_CREDIT_MODEL_SETTINGS number of credit model settings.

22.3.2.34 enum MOODYS_FEE_CAL_CODE

See Also

MOODYS_FEE_STRUCT

Enumerator

FEES_CALC_NONE	No fee calculation type will be applied.
FEES_TOTAL_POOL_CALC	Total pool fee.
FEES_TOTAL_BONDS_CALC	Total bond fee.
FEES_FIXED_AMOUNT_CALC	Fixed amount fee.
FEES_CUSTOM_CALC	Custom fee.

22.3.2.35 enum MOODYS_RATING_TYPE

See Also

set_whole_loan()

New feature Subject to change

Enumerator

MOODYS_RATING_Aaa	Aaa.
MOODYS_RATING_Aa1	Aa1.
MOODYS_RATING_Aa2	Aa2.
MOODYS_RATING_Aa3	Aa3.
MOODYS_RATING_A1	A1.
MOODYS_RATING_A2	A2.
MOODYS_RATING_A3	A3.
MOODYS_RATING_Baa1	Baa1.
MOODYS_RATING_Baa2	Baa2.
MOODYS_RATING_Baa3	Baa3.
MOODYS_RATING_Ba1	Ba1.
MOODYS_RATING_Ba2	Ba2.
MOODYS_RATING_Ba3	Ba3.
MOODYS_RATING_B1	B1.
MOODYS_RATING_B2	B2.
MOODYS_RATING_B3	B3.
MOODYS_RATING_Caa1	Caa1.
MOODYS_RATING_Caa2	Caa2.
MOODYS_RATING_Caa3	Caa3.
MOODYS_RATING_Ca	Ca.
MOODYS_RATING_C	C.
MOODYS_RATING_D	D.
MOODYS_RATING_LD	LD.
MOODYS_RATING_WR	WR.
MOODYS_RATING_NULL	N/R.

22.3.2.36 enum MOODYS_SWAP_NOTIONAL_CODE

See Also

MOODYS_HEDGE_STRUCT

Enumerator

SWAP_FIXED_LOAN_NOTIONAL Fixed swap.

SWAP_FLOATING_LOAN_NOTIONAL Floating swap.

SWAP_ALL_ASSETS_NOTIONAL All loans swap.

SWAP_CUSTOM_NOTIONAL Custom swap.

SWAP_CAP_NOTIONAL Caps swap.

SWAP_FLOOR_NOTIONAL Floors swap.

22.3.2.37 enum MPA_ANALYSIS_PARAM

The stress range type for MPA

New feature Subject to change

See Also

set_mpa_stress_range()

Enumerator

MPA_ANALYSIS_PARAM_PREPAY Indicating set cap and floor for prepay.

MPA_ANALYSIS_PARAM_DEFAULT Indicating set cap and floor for default.

MPA_ANALYSIS_PARAM_SEVERITY Indicating set cap and floor for severity.

22.3.2.38 enum MPA_ANALYSIS_PARAM_OFFSET

The offset type(LTV or FICO) for MPA.

New feature Subject to change

See Also

set_mpa_offset()

Enumerator

MPA_ANALYSIS_PARAM_OFFSET_LTV Indicating set LTV offset.

MPA_ANALYSIS_PARAM_OFFSET_FICO Indicating set FICO offset.

22.3.2.39 enum MPA_ANALYSIS_TYPE

Analysis type for MPA.

New feature Subject to change

See Also

set_mpa_analysis_type()

Enumerator

MPA_LOSS_SIMULATION Runs MPA credit model with loss simulation analysis. Typically runs 10000+ economic paths.

MPA_MEDC_SINGLE_PATH Runs MPA credit model with single path. It is run of a given MEDC scenario. MEDC scenario is predefined and in-build economic scenario and can be specified using a scenario number. The scenario number should be specified using API set_current_mpa_scenario.

MPA_MEDC_SIMULATION Reserved for future use.

MPA_CUST_MEDC_SINGLE_PATH Runs MPA credit model with single path of customized scenario. It is single path run of a user defined economic scenario. The User can provide an economic scenario path using custom economy API set_mpa_custom_scenario.

MPA_CUST_MEDC_SIMULATION Runs MPA credit model with MEDC simulation of customized scenario. It is simulation type analysis running 10000+ economic paths anchored to a user provided economic scenario. A user must create the economic scenario and can provide the scenario using API set_mpa_-custom_scenario.

MPA_FIXED_ECO_SIMULATION Runs MPA credit model with fixed economy predefined simulation sce-nario. This analysis type executes a predefined and in-build MEDC scenario path and simulate behavior of individual loan for 10000+ times. The selected economic scenario remains fix and only behavior of each loan in the provided pool is simulated each time.The scenario number should be specified using API set_current_mpa_scenario.

MPA_FIXED_CUST_SIMULATION Runs MPA credit model with fixed user defined simulation of customized scenario. This analysis type executes user defined scenario path and simulate behavior of individual loan for 10000+ times. The user defined scenario remains fix and only behavior of each loan in the provided pool is simulated each time. The User can provide an economic scenario path using API set_mpa_-custom_scenario.

NUM_MPA_ANALYSIS_TYPE Number of MPA analysis type.

22.3.2.40 enum MPA_MULTIPLIER_TYPE

Multiplier type for MPA

New feature Subject to change

See Also

set_mpa_multiplier()

Enumerator

MPA_MULTIPLIER_PREPAY Prepay multiplier.

MPA_MULTIPLIER_DEFAULT Default multiplier.

MPA_MULTIPLIER_SEVERITY Severity multiplier.

NUM_MPA_MULTIPLIER_TYPE Number of MPA multiplier type.

22.3.2.41 enum NON_PERFORMING_STATUS

Non performing loans status array index.

New feature Subject to change

See Also

set_default_non_performing_loans

22.3.2.42 enum OAS_CAL_MODE

See Also

- set_metrics_input_ex()

Enumerator

ENABLE_NONE Disable OAS calculation.

OAS_ONLY Just enable calculation of OAS, but disable calculation of effective duration, effective convexity.

ENABLE_ALL Enable calculation of OAS, effective duration, effective convexity.

22.3.2.43 enum PA_ANALYSIS_TYPE

Analysis type for PA.

New feature Subject to change

See Also

set_pa_analysis_type()

Enumerator

PA_LOSS_SIMULATION Standard Loss Simulation.

PA_MEDC_SINGLE_PATH Runs PA credit model with single path.It is run of a given MEDC scenario.MEDC scenario is predefined and in-build economic scenario and can be specified using a scenario number. The scenario number should be specified using API set_current_pa_scenario.

PA_CUST_MEDC_SIMULATION Runs PA credit model with MEDC simulation of customized scenario. It is simulation type analysis running 10000+ economic paths anchored to a user provided economic scenario. A user must create the economic scenario and can provide the scenario using API set_pa_custom_-scenario.

PA_CUST_MEDC_SINGLE_PATH Runs PA credit model with single path of customized scenario.It is single path run of a user defined economic scenario. The User can provide an economic scenario path using custom economy API set_pa_custom_scenario.

PA_FIXED_ECO_SIMULATION Runs PA credit model with fixed economy predefined simulation scenario. The scenario number should be specified using API set_current_pa_scenario.

PA_FIXED_CUST_SIMULATION Runs PA credit model with fixed user defined simulation of customized scenario. The User can provide an fixed economic scenario using API set_pa_custom_scenario.

NUM_PA_ANALYSIS_TYPE Number of PA analysis type.

22.3.2.44 enum PAYMENT_FREQUENCY

Payment frequency types

New feature Subject to change

See Also

GLOBAL_REINVESTMENT_ASSET_INFO

Enumerator

PAY_FREQ_MONTHLY Pay every month.

PAY_FREQ_SANNUALLY Pay every 6 months.
PAY_FREQ_QUARTERLY Pay every 3 months.
PAY_FREQ_ANNUALLY Pay every year.
PAY_FREQ_BIMONTHLY Pay every 2 months.
NUM_PAYMENT_FREQUENCY Number of payment frequency.

22.3.2.45 enum POOL_DELINQ_STATES_EX

Additional delinquent states

POOL_DELINQ_STATES_EX is the extension to POOL_DELINQ_STATES (In indextypes.h).

Enumerator

POOL_DELINQ_STATE_EX_BASE Place holder.
POOL_DELINQ_120P Delinquent >= 120 days.
POOL_DELINQ_150P Delinquent >= 150 days.
POOL_DELINQ_180P Delinquent >= 180 days.
POOL_DEFEASED Defeasance status.
POOL_NON_PERFORMING_MATURED_BALLOON Non performing matured balloon.
POOL_DELINQ_0P Delinquent >= 0 days.
POOL_BANKRUPT Bankrupt.
POOL_PAID_OFF Paid off.
POOL_REPURCHASED Repurchased.
POOL_LIQUIDATED Liquidated.
POOL_CLOSED Closed.
POOL_DELINQ_STATES_EX_SIZE Number of extra delinquency states.

22.3.2.46 enum POOL_DOCUM_TYPES_EX

Additional doc types

POOL_DOCUM_TYPES_EX is the extension to POOL_DOCUM_TYPES (In indextypes.h).

Enumerator

POOL_DOCUM_TYPES_EX_BASE Place holder.
POOL_DOCUM_FULL_ASSETS_PARTIAL_INCOME Documentation type: Full Assets - Partial Income.
POOL_DOCUM_FULL_INCOME_NO_ASSETS Documentation type: Full Income - No Assets.
POOL_DOCUM_NO_INCOME_PARTIAL_ASSETS Documentation type: No Income - Partial Assets.
POOL_DOCUM_NO_INCOME_STATED_ASSETS Documentation type: No Income - Stated Assets.

22.3.2.47 enum POOL_PROPERTY_TYPES_EX

Additional pool property types

POOL_PROPERTY_TYPES_EX is the extension to POOL_PROPERTY_TYPES (In indextypes.h).

Enumerator

POOL_PROPERTY_TYPE_EX_BASE Place holder.
POOL_PROPERTY_MULTI_FAMILY_TWO Property Type: Two Family.
POOL_PROPERTY_MULTI_FAMILY_THREE Property Type: Three Family.
POOL_PROPERTY_MULTI_FAMILY_FOUR Property Type: Four Family.
POOL_PROPERTY_MULTI_FAMILY_FIVEPLUS Property Type: Five or more Family.

22.3.2.48 enum PREPAY_DEFAULT_CALC_METHOD_TYPE

Prepayment and default calculation methods

See Also

- set_calculation_method()
- get_calculation_method()

Note

- Def[i] means defaults at period i of cashflow
- Ppy[i] means prepayments at period i of cashflow
- SchdPrin[i] means scheduled principal at period i of cashflow
- OrigAmort[i] means performing balance at period i of the original amortization schedule with no prepayments and defaults
- SMM[i] means single monthly mortality default rate at period i

Enumerator

PREPAY_DEFAULT_BEFORE_SCHED_PRIN_PPYDEF Default & Prepay before Scheduled Principal

Default, Prepayment and scheduled principal are calculated by the following formula:

- Def[i] = PerfBal[i-1] ∗ SMM[i]
- Ppy[i] = PerfBal[i-1] ∗ OrigAmort[i] / OrigAmort[i-1] ∗ SMM[i]
- SchdPrin[i] = (PerfBal[i-1] - Def[i]) ∗ (1 - OrigAmort[i] / OrigAmort[i-1])

SCHED_PRIN_PREPAY_BEFORE_DEFAULT_PPYDEF Scheduled Principal & Prepay before Default

Default, Prepayment and scheduled principal are calculated by the following formula:

- SchdPrin[i] = PerfBal[i-1] ∗ (1 - OrigAmort[i] / OrigAmort[i-1])
- Ppy[i] = PerfBal[i-1] ∗ OrigAmort[i] / OrigAmort[i-1] ∗ SMM[i]
- Def[i] = (PerfBal[i-1] - SchdPrin[i] - Ppy[i]) ∗ SMM[i]

DEFAULT_BEFORE_SCHED_PRIN_PREPAY_PPYDEF Default before Scheduled Principal & Prepay

Default, Prepayment and scheduled principal are calculated by the following formula:

- Def[i] = PerfBal[i-1] ∗ SMM[i]
- SchdPrin[i] = (PerfBal[i-1] - Def[i]) ∗ (1 - OrigAmort[i] / OrigAmort[i-1])
- Ppy[i] = (PerfBal[i-1] - Def[i] - SchdPrin[i]) ∗ SMM[i]

DEFAULT_PREPAY_SCHED_PRIN_PPYDEF Default, Prepay, Scheduled Principal

Default, Prepayment and scheduled principal are calculated by the following formula:

- Def[i] = PerfBal[i-1] ∗ SMM[i]
- Ppy[i] = (PerfBal[i-1] - Def[i]) ∗ SMM[i]
- SchdPrin[i] = (PerfBal[i-1] - Def[i] - Ppy[i]) ∗ (1 - OrigAmort[i] / OrigAmort[i-1])

JAPANESE_PREPAY_DEFAULT_PPYDEF Japanese default and prepay convention

Default, Prepayment and scheduled principal are calculated by the following formula:

- Def[i] = PerfBal[i-1] ∗ SMM[i]
- Ppy[i] = PerfBal[i-1] ∗ OrigAmort[i] / OrigAmort[i-1] ∗ SMM[i]
- SchdPrin[i] = (PerfBal[i-1] - Def[i]) ∗ (1 - OrigAmort[i] / OrigAmort[i-1])

  Note

  For student loan deals, "JAPANESE_PREPAY_DEFAULT_PPYDEF" is an invalid input.

NUM_PREPAY_DEFAULT_METHODS Max methods num.

22.3.2.49 enum PREPAY_DEFAULT_COMPOUNDING_METHOD

Prepayment compounding methods.

See Also

- set_prepay_default_compounding_method()

Enumerator

PREPAY_DEFAULT_COMPOUNDING_MONTHLY Prepayment compounding bases on monthly.

PREPAY_DEFAULT_COMPOUNDING_PERIODICITY Prepayment compounding bases on asset's periodicity.

22.3.2.50 enum RATING_AGENCY

Types of rating agency.

See Also

get_bond_rating_by_tranche()

22.3.2.51 enum REINV_OVERRIDE_TYPE

Reinvestment override types

New feature Subject to change

See Also

set_pv_reinvest_override

Enumerator

REINV_OVERRIDE_ALWAYS Test is not considered for Auto Reinvestment. This is the default.

REINV_OVERRIDE_NEVER If any test labeled 'Never' is failing, Auto Reinvestment is not active until all 'Never' tests pass again.

REINV_OVERRIDE_REINV_PER This is treated as 'Always' during the RP, 'Never' during After RP.

NUM_REINV_OVERRIDE_TYPE Number of reinvestment override types.

22.3.2.52 enum REINV_TERM_SETTING_TYPE

Reinvestment term setting types

New feature Subject to change

See Also

GLOBAL_REINVESTMENT_INFO

Enumerator

TERM_NON_SEASONED_INPUT The value to the term of an asset will not be adjusted by the age of the deal.

TERM_SEASONED_INPUT The value to the term of an asset will be adjusted by the age of the deal.

TERM_WAL_CURRENT The value to the term of an asset will be overriden by the weighted average life of current collateral pool.

TERM_WAL_TRIGGER The value to the term of an asset will be overriden by the value of the trigger of the weighted average life quality test.

NUM_TERM_SETTING_TYPE Number of term setting types.

22.3.2.53 enum REINV_TYPE

Reinvestment types.

New feature Subject to change

See Also

set_reinvestment_type()

Enumerator

DEAL_REINV Use the deal level reinvestment settings.

GLOBAL_REINV Use global reinvestment settings.

NUM_REINV_TYPE Number of reinvestment types.

22.3.2.54 enum RESEC_EXCEPTIONS_HANDLING

Resec Exception Handling

New feature Subject to change

See Also

set_resec_exceptions_handling

Enumerator

RESEC_EXCEPTIONS_HANDLING_TREAT_AS_NONRESEC Treat missing underlying deals or circularly referenced deals as non-resec.

RESEC_EXCEPTIONS_HANDLING_TREAT_AS_ERROR Treat missing underlying deals or circularly referenced deals as errors.

NUM_RESEC_EXCEPTIONS_HANDLING_TYPE Number of resec exceptions handling types.

22.3.2.55 enum SCENARIO_RATE_SHIFT_TYPE

See Also

set_rate_shift_setting

22.3.2.56 enum SEASONING_TYPE

Enumerator

SEASONING_GLOBAL Take into account the update month of the deal, then start all loans at that point of the vector. For example, if the deal is in its 12th payment month, everything will run at month 12 of the vector from today forward.

SEASONING_NO Do not apply any seasoning to the deal, so all loans will start at the first month of the vectors.

SEASONING_YES Apply loan level seasoning, so every loan backing the deal will start at the point of the vector corresponding to that loan's seasoning.

22.3.2.57 enum SERVICER_ADVANCES_BASE

See Also

set_service_advances_rates_type()

New feature Subject to change

Enumerator

SERVICER_ADVANCES_BASE_OFF The servicer advance projections are disable.

SERVICER_ADVANCES_BASE_DELINQ Indicate vector/constant is projected against delinquent balance whose projection is set by set_addit_group_delinquencies().

SERVICER_ADVANCES_BASE_DEFAULT Indicate vector/constant is projected against default balance.

22.3.2.58 enum SIMULATION_TYPE

Simulation engine type

New feature Subject to change

See Also

set_simulation_engine

Enumerator

SIMULATION_MONTE_CARLO Monte carlo.

SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION Default probability distribution.

NUM_SIMULATION_TYPE Number of simulation types.

22.3.2.59 enum TEST_TYPE

CDO test type

New feature Subject to change

Enumerator

TEST_IC Interest coverage test.

TEST_PV Par value test.

TEST_UD User defined test.

22.3.2.60 enum UK_REGION

See Also

set_whole_loan()

New feature Subject to change

Enumerator

IGBR	UK.
EAMI	East Midlands.
EAST	Eastern.
LOND	London.
NEAS	North East.
NORW	North West (including Merseyside)
NOIR	Northern Ireland.
SCTL	Scotland.
SOEA	South East.
SOWE	South West.
WALS	Wales.
WEMI	West Midlands.
YOHU	Yorkshire and The Humber.

22.3.2.61 enum US_STATE

See Also

set_whole_loan()

New feature Subject to change

22.3.2.62 enum WHOLE_LOAN_AMORTIZATION_TYPE

Enumerator

ANN	Annuity amortization schedule.
LIN	Straight line amortization schedule.
BUL	Bullet amortization schedule.
BULINV	Bullet and Investment.

22.3.2.63 enum WHOLE_LOAN_COUPON_TYPE

See Also

set_whole_loan()

New feature Subject to change

Enumerator

FIXED_COUPON	Fixed coupon;.

FLOATING_COUPON	Floating coupon;.
FIXED_TO_FLOATING	Fixed to floating coupon;.
FLOATING_TO_FIXED	Floating to fixed coupon;.
FIXED_STEP	Fixed step coupon;.
FLOATING_STEP	Floating step coupon;.
FIXED_TO_FLOATING_STEP	Fixed to floating step coupon;.
FLOATING_TO_FLOATING_STEP	Floating to floating step coupon;.

22.3.2.64 enum WHOLE_LOAN_DEFAULT_METHOD_TYPE

New feature Subject to change

22.3.2.65 enum WHOLE_LOAN_ISSUER_TYPE

See Also

set_whole_loan()

New feature Subject to change

Enumerator

WL_FNMA	Issued by Fannie Mae;.
WL_FHLMC	Issued by Freddie Mac;.
WL_GNMA	Issued by Ginnie Mae;.
WL_NA_SUBPRIME	Non-agency, Subprime underwriting;.
WL_NA_PRIME	Non-agency, Prime underwriting;.
WL_ISSUER_OTHER	All others & unknown underwriting;.

22.3.2.66 enum WHOLE_LOAN_OCCUPANCY_TYPE

Enumerator

WL_OCCUPANCY_UNK	Unknown.
WL_OCCUPANCY_OWN	Owner-occupied/Primary.
WL_OCCUPANCY_SEC	Holiday/second home.
WL_OCCUPANCY_INV	Non-owner-occupied/buy-to-let/Investment.

22.3.2.67 enum WHOLE_LOAN_TYPE

See Also

set_whole_loan()

New feature Subject to change

Enumerator

WL_RMBS	RMBS loan type;.
WL_ABS_AUTO	ABS Auto loan type;.

WL_ABS_STUDENT_LOAN ABS student loan loan type;.

WL_ABS_CREDIT_CARD ABS credit card loan type;.

WL_ABS_AUTO_LEASE ABS auto lease loan type;.

WL_CMBS CMBS loan type;.

WL_CDO CDO loan type;.

WL_HECM HECM loan type;.

WL_REVERSE_MORTGAGE Reverse mortgae loan type;.

WL_TYPE_OTHER other loan type;

22.3.3 Function Documentation

22.3.3.1 int CHASAPI adjust_PA_vectors (void * *tid,* bool *enable*)

This method is adjust PA vectors for SFW student loan deals.

Since

3.6

Availability SFW

Precondition

open_deal_ex(), set_moodys_credit_model_settings() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	No error.
-1	Error - Deal not open.
-2	Error - Not set PA credit model.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*
*       assert(0 == adjust_PA_vectors(pDeal, true));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

This function only be called after set PA credit model.

22.3.3.2 int CHASAPI calculate_bond_first_loss (void * *tid,* const char * *bondid,* FIRST_LOSS_INPUT *first_loss_input,* FIRST_LOSS_RESULT * *first_loss_result*)

This method exposes first loss calculator functionality in WSAAPI

Since

3.4.1

Availability SFW, CDOnet

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
in	*first_loss_input*	Input information for calculating bond first loss.
out	*first_loss_result*	A pointer to the result of calculate bond first loss.

Return values

0	No error.
-1	Error - Deal not open.
-2	Error - Invalid Input for first_loss_input.first_loss_run_mode.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "TCCT-II");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*       FIRST_LOSS_INPUT first_loss_input;
*       first_loss_input.first_loss_run_mode =
      FIRST_LOSS_RUN_MODE_LOSS_RATE;
*       first_loss_input.is_percentage = false;
*       first_loss_input.prepayment_type = PREPAY_CURVE_CPR;
*   first_loss_input.prepayment_rate = 0.0;
*
*       first_loss_input.default_type = DEFAULT_CURVE_CDR;
*   first_loss_input.default_rate = 0.0;
*   first_loss_input.forbearance_rate = 0.0;
*   first_loss_input.deferment_rate = 0.0;
*   first_loss_input.first_loss_threshold =
      FIRST_LOSS_THRESHOLD_INTEREST;
*
*   first_loss_input.principal_payment_rate = 0;
*   first_loss_input.monthly_purchase_rate = 0;
*   first_loss_input.portfolio_yield = 0;
*   first_loss_input.loss_rate = 0;
*
*   first_loss_input.principal_payment_rate = 0;
*   first_loss_input.prin_loss_serverity = 0.0;
*
*   FIRST_LOSS_RESULT first_loss_result;
*      assert(0 == calculate_bond_first_loss(pDeal, "1601-A", first_loss_input, &
      first_loss_result));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

For CDONET reremic deal, this function only takes top deal into calculation.

22.3.3.3 long CHASAPI clear_moodys_credit_model_setttings (void ∗ *tid*)

Clear sfw credit model

Since

1.4.0

Availability SFW, CHS

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strncpy(pCmo->settlement_date,"04/01/14",9);
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
      PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
*      open_deal_ex(tid,pCmo);
*
*      set_moodys_credit_model_settings(tid,
      MOODYS_PA_SETTINGS, false);
*      set_pa_default_pool_data(tid,"PurposePurchase","100");
*      set_pa_default_pool_data(tid,"PurposeRefi","0");
*      set_pa_default_pool_data(tid,"OccupancyOwner","50");
*      set_pa_default_pool_data(tid,"OccupancySecondHome","50");
*      set_pa_default_pool_data(tid,"OccupancyInvestor","0");
*      set_pa_default_pool_data(tid,"Property1Unit","100");
*      set_pa_default_pool_data(tid,"Property24Unit","0");
*      set_pa_default_pool_data(tid,"OriginatorThirdParty","50");
*      set_pa_default_pool_data(tid,"OriginatorRetail","50");
*      set_pa_default_pool_data(tid,"HARP1","50");
*      set_pa_default_pool_data(tid,"HARP2","50");
*      set_pa_default_pool_data(tid,"FHA","100");
*      replace_pa_pool_data(tid,1,"WAFICO","300");
*
*      run_deal_ex(tid,pCmo);
*
*      MARKIT_COLLAT_CASHFLOW collCashflow;
*      memset(&collCashflow, 0, sizeof(MARKIT_COLLAT_CASHFLOW) );
*      get_collateral_flow_ex1(tid, 0, 0, &collCashflow);
*
*      for(int i=1;i<collCashflow.size-2;i++)
*      {
*          EXPECT_GT(collCashflow.prepayments[i],0.0);
*      }
*
*      clear_moodys_credit_model_setttings(tid);
*
*      double CPR = 0;
*      set_prepayments_ex(tid, PREPAY_CURVE_CPR, 0, &CPR, -1, true);
*      // run standard mode.
*      run_deal_ex(tid,pCmo);
*
*      close_deal_ex(tid, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.4 long CHASAPI enable_bond_insurance (void ∗ *tid,* const char ∗ *bondid,* BOOLYAN *is_enabled*)

Enable bond insurance.

Since

3.0.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
in	*is_enabled*	The flag indicate whether this bond is insured.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid bondid.
-99	Error - For details call get_deal_error_msg()

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT;
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "SAS059XS");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      int ret = open_deal_ex(tid, pCmo);
*      if(ret < 0)
*      {
*          //Error handling
*      }
*
*      const int len = 1024;
*      char input_csv[len]={0};
*      char output_csv[len]={0};
*      ret = get_cmm_input_files(tid,NULL,input_csv,len,output_csv,len);
*      if(ret < 0)
*      {
*        //Error handling
*      }
*
*      ret = enable_bond_insurance(tid, "1-A3A", true);
*
*      ret = close_deal_ex(tid, pCmo);
*      if(ret < 0)
*      {
*          //Error handling
*      }
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.5 int CHASAPI enable_default_on_snapshot_date (void * *tid*, BOOLYAN *flag_snapshot*)

Enable Snapshot Date Logic in Whole Loan Analyzer loan cashflow projection, which impacts the order of loan payments. The snapshot dates are the dates on which the engine snapshots the state of the whole portfolio. The snapshot dates are the dates monthly incremental based on settlement date set in set_whole_loan(). Snapshot date logic assumes:

- Default events, including default, loss and recovery, can only happen on snapshot day of each month, regardless of loan pay frequency.
- All other payments happen on loan payment dates.

- If snapshot day and loan payment day are the same, default event goes first.

Since

3.7.0

Availability SFW

Precondition

set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*flag_snapshot*	The flag of using snapshot date logic or not.

Return values

0	Success.
-1	Error - Set_whole_loan() not called.
-4	Error - Other error.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       //set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*       enable_default_on_snapshot_date(pDeal, true);
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.6 int CHASAPI enable_periodic_coupon_rate_projection (void * *tid*, BOOLYAN *flag_periodic_rate*)

Use the actual coupon rate adjusted by day calendar to do cashflow amortization in whole loan analyzer.

Since

3.4.0

Availability SFW

Precondition

set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

in	*flag_periodic_-rate*	If TRUE the projection would use the periodic coupon rate.

Return values

0	No error.
-1	Error - set_whole_loan() was not called.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* pDeal = NULL;
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       //set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*       assert(0 == enable_periodic_coupon_rate_projection(pDeal, 1));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

If this set to true, the loan coupon will be modified in pay reset month.

22.3.3.7 int CHASAPI enable_reinv_loan (void * *tid,* BOOLYAN *populate_reinv_loan*)

This method sets whether to expose the loans with zero face value in "REINV" pool group. If input parameter use-_zero_loan is TRUE, API will read not only the loans with non-0 face value, but also loans with zero face value in "REINV" pool group; if it is FALSE, API will only read the loans with non-0 face value.

Since

3.0.0

Availability CDONET

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*populate_reinv_-loan*	If use_zero_loan is set to TRUE, loans with zero face value in "REINV" pool group will be read. If use_zero_loan is set to FALSE, only loans with non-0 face value will be read.

Return values

0	Success.
-1	Error - Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "1776");
*
*       set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == enable_reinv_loan(pDeal, true));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.8 long CHASAPI enable_sfw_delinq_projection (void ∗ *tid*, BOOLYAN *is_enabled*)

This function will enable sfw delinquency projection.

Since

3.0.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_enabled*	The flag indicate whether using default delinquency assumption.

Return values

0	Success.
-1	Error - Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == enable_sfw_delinq_projection(pDeal, 1));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.9 int CHASAPI generate_cmm_custom_result_output (void ∗ *tid*, char ∗ *custom_scen_name*)

Generates the CMM custom scenario output file, it would call the cmm custom callback function to generate the CMM custom scenario output file.

Since

3.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to CMM with API set_moodys_credit_model_settings().
set_cmm_custom_scenario() has been called.
SetupCMMCustomModel() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*custom_scen_-name*	A pointer to name of the CMM custom scenario. It is defined by user.

Return values

0	Success
-1	Error - Deal not opened
-2	Error - ME or IR data not existed(set in set_cmm_custom_scenario())
-3	Error - CMM Loan data not existed.
-99	Error - For details call get_deal_error_msg()

Example:

```
*       #include "MarkitCMMProvider/MarkitCMMProvider.h"
*       #pragma comment(lib, "WSACMMProvider.lib")
*
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       int ret = open_deal_ex(pDeal, pCmo);
*       if(ret < 0)
*       {
*           //Error handling
*       }
*
*       set_moodys_credit_model_settings(pDeal,
       MOODYS_CMM_SETTINGS, false);
*       double value[] = {0.056999998,0.056258359,0.054541469,0.056365418,0.064670191,0.0782126,0.086426401,
       0.090757341,0.091721907,0.09174448,0.091839638,0.089244394,0.08650013,0.084163132,0.079800591,0.073180208,0.
       067184458,0.061098261,0.056628218,0.05455205,0.054580388,0.0547897323333333,0.0549990766666667,0.055208421,0
       .055474019,0.055556188,0.055709429,0.055890002,0.05600625,0.055872004,0.055737758,0.055525441,0.055487661,0.
       055262222,0.055346231,0.055127888,0.055127888,0.054981022,0.054845071,0.054779229};
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_REALGDPGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_UNEMPRATE, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_FEDFUNDSRATE, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_TSY10Y, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_CPIINFRATE, value, 40);
*
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_POPGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_NUMHOUSEHOLDSGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_RETAILSALESGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_TOTNONFARMEMPGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_NOMPERSONALINCGROWTH, value, 40);
*
```

```
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
      CMM_ME_HOMEPRICEGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
      CMM_ME_BAACORPYIELD, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
      CMM_ME_CREPXIDXGROWTH, value, 40);
*
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_IR,
      CMM_IR_LIBOR1M, value, 40);
*
*       char* custom_scen_name = "scenario1";
*
*       SetupCMMCustomModel(pDeal, "user", "123456");
*       set_current_moodys_cmm_scenario(pDeal, NULL, custom_scen_name);
*
*       generate_cmm_custom_result_output(pDeal, custom_scen_name);
*       RemoveCMMCustomModel(pDeal);
*
*       run_deal_ex(pDeal,pCmo);
*       close_deal_ex(pDeal,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

Generating CMM custom scenario file need the cmm input csv file, please make sure cmm data file have been downloaded.

22.3.3.10 int CHASAPI generate_forward_interest_rates (void * *tid*)

Generate forward interest rate curves from user-input market rates.

Since

3.0.0

Availability SFW, CDOnet, CHS

Precondition

set_index_rate(), load_MWSA_rates() or set_rate_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	Success.
-1	Insufficient market rate information for [currency] [rate type]. Minimum inputs are [...].
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "CQSCLO2");
*
*       open_deal_ex(tid, pCmo);
*
*       short idx = LIBOR_1;
*       double rate = 0.00853;
*       set_index_rate(pDeal, "USD", &idx, 0, &rate);
*
```

```
*       generate_forward_interest_rates(tid);
*       double *pRate = get_forward_interest_rates(tid, "USD", &idx);
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.11 int CHASAPI get_asset_type_list (char ∗ *asset_type_list[ ],* char ∗ *err_buffer,* int *err_length*)

Get asset type list from WSAAPI.DB.

Since

3.1.0

Availability SFW, CHS, CDONET

Parameters

out	*asset_type_list.*	A client-allocated char array for the asset type information which will be stored, Null if first call.
out	*err_buffer.*	Buffer to get error message content.
in	*err_length.*	The length of err_buffer defined by user.

Return values

>=0	Actual number of asset type returned.
-1	Error - DB not found.
-99	Error - Please examine err_buffer for error.

Example:

```
*       char err_buffer[200] = "";
*       int err_length = 200;
*       int asset_number = get_asset_type_list(NULL, err_buffer, &err_length);  // first
        call to get number of asset type
*       if (asset_number > 0)
*       {
*           std::vector<char*> asset_type_list(asset_number);
*           std::vector<char> asset_type_list_buf(asset_number*200);
*           for(int i = 0; i < asset_number; i++) asset_type_list[i] = &asset_type_list_buf[i*200];
*           get_asset_type_list(&asset_type_list.front(), err_buffer, err_length);  //
        second call to get asset_type_list
*       }
*
```

Note

Pass NULL for asset_type_list to get the number of asset type on first call. And then get asset_type_list on second call.

22.3.3.12 long CHASAPI get_available_borrower_benefits (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* BORROWER_BENEFIT_ELIGIBILITY *benefit_list[ ],* int *size*)

Retrieves the list of borrower benefits for SLABS deals

This function allows users to retrieve the full list of borrower benefits available to SLABS deals.

New feature Subject to change

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	Reremic deal id for a child deal, otherwise null.
out	*benefit_list*	A pointer to an array which stores the information of the available borrower benefits of a deal.
in	*size*	size of the benefit_list that user has passed.

Return values

>=0	The number of borrower benefits in deal.
-1	Error - Deal not opened
-2	Error - Invalid size
-5	Error - Current deal is not SLABS
-99	Error - For details call get_deal_error_msg()

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0 ,sizeof(*pCmo) );
*       strcpy(pCmo->dealid, "AB05HE3");
*
*       open_deal_ex(tid,pCmo);
*
*       get_available_borrower_benefits(tid, NULL, NULL, 0);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.13 int CHASAPI get_balloon_extension_assumptions (void * *tid,* const char * *reremic_deal_id_or_null,* int * *months,* double * *rates,* int *length,* int * *delay,* long *loan_num*)

Get the balloon extension assumption used for the specified piece of collateral.

New feature Subject to change

Since

2.6.5

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	If reremic deal, this is the id, otherwise null
out	*months*	A pointer to an array which stores the months that the user set for each balloon extension period.
out	*rates*	A pointer to an array which stores the rates that the user set for calculating the extension penalties for each balloon extension period
in	*length*	The number of extensions set, up to 3 extensions
out	*delay*	A pointer to an int which sotres the number of months delayed as the extension penalties are paid.
in	*loan_num*	The 1-based index of the loan in the deal.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid input length
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Note

For CMBS only

Example:

```
*        void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "CMBS_CCC070C3");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       int pblMonths[3]={0};
*       double pblRates[3] = {0};
*       int blLength = 3;
*       int pblDelay[1] = {0};
*       long loan_num = 94;
*       int ret = get_balloon_extension_assumptions(pDeal,NULL, pblMonths,
       pblRates,blLength,pblDelay,loan_num);
*       if(ret < 0)
*       {
*           //error handling
*       }
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.14 int CHASAPI get_bond_authorized_integral_amount (void ∗ *tid,* char ∗ *bondid,* double ∗ *value*)

Since

3.1.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The bond name.
out	*value*	Authorized integral amount for specific bondid.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*   void *pDeal = NULL;
*   CMO_STRUCT cmo;
*   memset(&cmo, 0, sizeof(CMO_STRUCT));
*   strcpy(cmo.dealid, "STATICLO");
*
*   set_engine_preference(PICK_SFW_ENGINE_FOR_MAPPED_DEALS
      );
*   open_deal_ex(pDeal, &cmo);
*
*   double value;
*   int ret = get_bond_authorized_integral_amount(pDeal, "A", &value);
*   if (0 != ret)
*   {
*           // error handle
*   }
*
*   close_deal_ex(pDeal, &cmo);
*
```

22.3.3.15 int∗ CHASAPI get_bond_cf_dates (void ∗ *tid,* const char ∗ *bondid*)

This method returns the cash flow dates of a specified bond.

Since

2.1.0

Availability CDOnet, SFW, CHS

Precondition

deal is run

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The NULL terminated string indicate the name of the tranche whose results are being requested.

Return values

>0	A pointer to an array which stores the dates requested. The size of the vector is MAX_PERIODS. The first period stores the last payment date of the current deal update.

NULL	Cashflow is not available. Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
       SIMULATION_MONTE_CARLO));
*
*       // add settings for simulation and run simulation
*
*       assert(NULL != get_bond_cf_dates(pDeal, "A2")); // call after run simulation
*       assert(NULL != get_coll_cf_dates(pDeal));       // call after run simulation
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.16 int CHASAPI get_bond_cf_length (void ∗ *tid,* short *path,* const char ∗ *bondid*)

This function returns the length of available cash flows of a specified bond in a specified run.

Since

2.1.0

Availability CDOnet, SFW, CHS

Precondition

open_deal_ex() has been called.
run_deal_ex() or run_monte_carlo_simulation() or run_default_probability_distribution() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*path*	The index of the path whose bond cashflows are being requested, 0 for the average bond cashflows or for the bond cashflow of run_deal_ex(). CHS currently only supports the bond cashflow of run_deal_ex().
in	*bondid*	Name of the tranche whose results are being requested. The length should be 20.

Return values

>=0	Success. Return the length of available cash flows of the specified bond in the specified run.
-1	Deal not opened.
-2	Error - Bond id not recognized.
-3	Error - Current simulation engine is not set to any available one.
-4	Error - Invalid path.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ACE06NC1");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*      assert(0 == run_monte_carlo_simulation(pDeal));
*
*      assert(get_bond_cf_length(pDeal, 1, "A1") > 0);
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.17 int CHASAPI get_bond_currency (void ∗ *tid,* const char ∗ *bondid,* char ∗ *currency*)

This method get the currency code for a specified bond .

Since

2.5.0

Availability All

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The specified bond id.
out	*currency*	A pre-allocated pointer of at least 4 characters for the currency code of the specified bond.

Return values

0	Success.
-1	Deal not open.
-2	Invalid bondid.
-3	currency pointer is null.
-99	Error ,call get_deal_error_msg() for details.

Example:

```
*     void *pDeal = NULL;
*     //deal has been opened
*
*     char currency[4] = {0};
*     int ret = get_bond_currency(pDeal, "A1", currency);
*     if (0 != ret)
*     {
*         // error handle
*     }
*
```

22.3.3.18 int CHASAPI get_bond_FFIEC_results (void ∗ *tid,* const char ∗ *bondid,* FFIEC_INPUT_PARAMS ∗ *FFIEC_inputs,* FFIEC_RESULTS *FFIEC_results[]*)

Get the FFIEC test results for the specified bond

Since

3.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
run_FFIEC_test() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
in	*FFIEC_inputs*	FFIEC input information, if NULL, FFIEC test results would base on NO_BENCH_MODE return.
out	*FFIEC_results*	A user allocated array which stores the FFIEC results returned, the array length must be at least 7.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_msg()

Note

Current number of FFIEC scenarios is 7,1∼7 means interest rate +300, +200, +100, +0, -100, -200 , -300bps. The FFIEC_results input array length must be at least 7.

Example:

```
*       void* pDeal = NULL;
*       //deal has been opened.
*
*       run_FFIEC_test(pDeal, PREPAY_CURVE_SMM, NULL);
*       FFIEC_RESULTS FFIECResults[7];
*       memset(FFIECResults, sizeof(FFIEC_RESULTS) * 7);
*       get_bond_FFIEC_results(pDeal, pCmo->bond.
       stripped_id, NULL, FFIECResults);
*
```

22.3.3.19 double∗ CHASAPI get_bond_flow_sim (void ∗ *tid,* short *path,* const char ∗ *bondid,* int *flow_identifier*)

This method returns the simulation bond cashflow calculated. The maximum for path number of cash flows populated is 100 by default. To remove the limit, please call function remove_simulation_cashflow_populated_limit.

Since

2.1.0

Availability CDOnet, SFW

Precondition

run_monte_carlo_simulation() or run_default_probability_distribution() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*path*	The path number of the requested bond cashflow.0 for the average bond cash-flows.
in	*bondid*	The name of tranche whose bond cashflow is being requested.
in	*flow_identifier*	The bond cashflow identifier being requested.

Return values

OTHER	Pointer to the vector of cashflows.
NULL	Error - Call get_deal_error_msg().

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*       MONTE_CARLO_ASSUMPTION basic_assumption;
*       // assign members of basic_assumption
*       MONTE_CARLO_DEF_PPY_REC_ASSUMPTION def_ppy_rec_assumption;
*       // assign members of def_ppy_rec_assumption
*       assert(0 == set_monte_carlo_assumption(pDeal, &basic_assumption, &
      def_ppy_rec_assumption));
*       assert(0 == run_monte_carlo_simulation(pDeal));
*
*       double* balance = get_bond_flow_sim(pDeal, 1, "A1",
      FLOW_BOND_BALANCE);
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.20 int CHASAPI get_bond_implied_loss (void ∗ *tid*, const char ∗ *bondid*, double ∗ *implied_loss*)

This method gets the implied loss for a bond.

Since

2.4.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
out	*implied_loss*	The implied loss value.

Return values

0	Success.
-1	Deal not open.
-2	Invalid pointer.
-3	Implied loss is "NOT AVAILABLE".
-10	Bond not found.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AL2010-A");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       double impliedLosses = 0.0;
*       assert(0 == get_bond_implied_loss(pDeal, "A1", &impliedLosses));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.21 int CHASAPI get_bond_info_by_index_ex (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int *index,* MOODYS_BOND_INFO ∗ *bond_info*)

Retrieves the additional desctiptive bond information by index

New feature Subject to change

Since

3.0.0

Availability CDOnet, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	Pass NULL for main deal or remic name for underlying deal.
in	*index*	The 1-based index of the bond in the array of bonds.
out	*bond_info*	Pointer to the structure holding bond info.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       MOODYS_BOND_INFO result;
*       assert(0 == get_bond_info_by_index_ex(pDeal, NULL, 1, &result));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.22 int CHASAPI get_bond_info_by_tranche_ex (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *bondid,* MOODYS_BOND_INFO ∗ *bond_info*)

Retrieves the additional desctiptive bond information by bondid

New feature Subject to change

Since

3.0.0

Availability CDOnet, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	Pass NULL for main deal or remic name for underlying deal.
in	*bondid*	A pointer to the name of the bond.
out	*bond_info*	Pointer to the structure holding bond info.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       MOODYS_BOND_INFO result;
*       assert(0 == get_bond_info_by_tranche_ex(pDeal, NULL, "A1", &result));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.23 int CHASAPI get_bond_market_risk_metrics (void * *tid*, const char * *bondid*, METRIC_INPUT_STRUCT * *metric_inputs*, METRIC_RESULTS_STRUCT * *metric_results*)

This method returns the market risk metrics calculation result of a specified bond.

Since

3.0.0

Availability ALL

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond..
in	*metric_inputs*	The inputs for calculating metrics result.
out	*metric_results*	The pointer of the market risk result.

Return values

=0	Success
-1	Deal not open.
-2	Error - Invalid bond name.
-3	Error - bond not found.
-4	Error - Invalid index type.
-5	Error - Invalid pointer of metric results.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      METRIC_INPUT_STRUCT metric_input;
*      memset(&metric_input, 0, sizeof(METRIC_INPUT_STRUCT));
*      metric_input.clean_price = 100;
*      metric_input.apply_spread_to = APPLY_SPREAD_TO_TSY;
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      METRIC_RESULTS_STRUCT results_m;
*      memset(&results_m, 0, sizeof(METRIC_RESULTS_STRUCT));
*      assert(0 == get_bond_market_risk_metrics(pDeal, bondid, &metric_input, &
      results_m));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.24 int CHASAPI get_bond_market_risk_metrics_ex (void * *tid*, char * *bondid*, METRIC_ANCHORS *anchor_type*, double *anchor_value*, APPLY_SPREAD_TYPE *apply_to*, METRIC_RESULTS_STRUCT_EX * *results_ex*)

This method returns the extra market risk metrics calculation result of a specified bond.

Since

3.0.0

Availability ALL

Precondition

set_metrics_input_ex() with OAS_CAL_MODE enabled has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
in	*anchor_type*	Anchor type for metrics calculation. Available options: MARKET_PRICE or OAS, if input MARKET_PRICE then return oas result, otherwise return market price.
in	*anchor_value*	Value of the provided metric anchor, corresponding to anchor_type in METRIC_INPUT_STRUCT_EX
in	*apply_to*	The type of metric calculation apply spread to TREATURY/LIBOR Curves.
out	*results_ex*	The pointer of the market risk result.

Return values

=0	Success
-1	Deal not open.
-2	Error - Invalid bond name.
-3	Error - bond not found.
-4	Error - Invalid anchor type.
-5	Error - Invalid spread apply type.
-6	Error - Invalid pointer of metric results.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      METRIC_RESULTS_STRUCT_EX metric_result_ex;
*      memset(&metric_result_ex, 0, sizeof(metric_result_ex));
*      assert(0 == get_bond_market_risk_metrics_ex(pDeal, bondid,
     MARKET_PRICE, 100, APPLY_SPREAD_TO_LIBOR, &metric_result_ex));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

If want to get the metric results of this method, set_metrics_input_ex must be called before.

Warning

In each path of OAS-related analysis, LIBOR 6 month spread input to MPA and PA is floored to 0.01% as required by MPA and PA; TSY 1 year and TSY 10 year input to MPA and PA is floored to -2% as required by MPA and PA.

22.3.3.25 int CHASAPI get_bond_next_reset_date (void * *tid,* const char * *bondid,* int * *next_reset_date*)

Expose next reset date.

Since

3.0.0

Availability SFW, CDOnet

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The NULL terminated string indicate the name of the tranche whose results are being requested.
out	*next_reset_date*	The value of next reset date, format "YYYYMMDD".

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Bond not found
-3	Error - Next reset date is not available
-99	Error - Other error

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT cmos;
*       memset(&cmos, 0, sizeof(cmos));
*       strcpy(cmo.dealid, "CQSCLO2");
*
*       assert(0 == open_deal_ex(tid, &cmos));
*       assert(0 == run_deal_ex(tid, &cmos));
*
*       int next_reset_date = 0;
*       assert(0 == get_bond_next_reset_date(tid, "B", &next_reset_date));
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.26 int* CHASAPI get_bond_payflag (void * *tid,* const char * *reremic_deal_id_or_null,* const char * *bondid*)

This method is to get bond payflag vector.

Since

2.7.0

Availability SFW,CDOnet

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*bondid*	The specified bond id.

Return values

NULL	Error - Call get_deal_error_msg().
OTHER	Pointer to the vector of cashflows.

Example:

```
*       void* pDeal= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SANGO1101P");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       // open deal
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       // run deal
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       // get pay flag
*       int* payflag = get_bond_payflag(pDeal, NULL, "A1");
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.27 int∗ CHASAPI get_bond_rate_reset_dates (void ∗ *tid,* const char ∗ *bondid*)

This method returns the array of rate reset date of a specified bond.

Since

2.7.0

Availability SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The NULL terminated string indicate the name of the tranche whose results are being requested.

Return values

>0	A pointer to an array which stores the bond rate reset dates requested. The size of the vector is 500.
NULL	reset date dates is not available. Call get_deal_error_msg() for detail.

Note

The rate reset dates just available for the floater bonds.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(NULL != get_bond_rate_reset_dates(pDeal, "A2"));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.28 int CHASAPI get_bond_rating_by_tranche (void ∗ *tid,* const char ∗ *bondid,* RATING_AGENCY *agency,* char ∗ *rating*)

This function will get a detail rating agency information for a bond.

Since

3.0.0

Availability CDOnet,SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
in	*agency*	Rating agency of bond. Should be one of RATING_AGENCY.
out	*rating*	The rating agency information. A pointer to the user allocated structure holding bond rating agency information.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid bondid.
-99	Error - For details call get_deal_error_msg()

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "1776");
*
*       set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*        char rating[5] = {0};
*        memset(rating, 0, sizeof(rating));
*       assert(0 == get_bond_rating_by_tranche(pDeal, "A2", MOODYS_CURRENT, rating
      ));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.29 int CHASAPI get_bond_step_up_coupon (void * *tid,* const char * *bondid,* BOND_STEP_UP_COUPON *all_set_up_coupons[ ],* int *array_size,* int * *num_available*)

This method get the bond step up coupon and related infomation for a specified bond.

Since

2.9.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The specified bond id.
out	*all_set_up_-coupons*	A pre-allocated array of client-allocated BOND_STEP_UP_COUPON struct for the step-up coupon of the specified bond.
in	*array_size*	Size of the array of BOND_STEP_UP_COUPON that user has passed.
out	*num_available*	Total number of step-up coupons for .

Return values

>=0	Success. Actual number of coupon for the specified bond.
-1	Deal not open.
-2	Invalid bondid.
-3	Bond not found.
-99	Error ,call get_deal_error_msg() for details.

Example:

```
 *      void *pDeal = NULL;
 *      //deal has been opened
 *
 *      int iret = get_bond_step_up_coupon(pDeal, "A" , NULL, 0, NULL);
 *      if (iret < 0)
 *      {
 *          // error handle
 *      }
 *
 *      BOND_STEP_UP_COUPON * pbond_stepup_coupon = (
        BOND_STEP_UP_COUPON*)malloc(sizeof(BOND_STEP_UP_COUPON) * iret);
 *      memset(pbond_stepup_coupon, 0, sizeof(BOND_STEP_UP_COUPON) * iret);
 *      int num_coupons = 0;
 *      iret = get_bond_step_up_coupon(pDeal, "A", pbond_stepup_coupon, iret, &
        num_coupons);
 *
```

22.3.3.30 int CHASAPI get_bond_total_loss (void * *tid,* const char * *bondid,* double * *total_loss*)

This method gets the total loss for a bond.

Since

4.0.0

Availability CDOnet, SFW, CHS

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
out	*total_loss*	The total loss value.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid pointer.
-3	Error - Bond not found.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg().

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "DRYDEN34");
*
*      assert(0 == open_deal_ex(tid, pCmo));
*      assert(0 == run_deal_ex(tid,pCmo));
*
*      const char* bondid = "1A1";
*       double total_loss = 0.;
*      int nRet = get_bond_total_loss(tid, bondid, &total_loss);
*      if(nRet !=0)
*      {
*           //error handle;
*      }
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.31 int CHASAPI get_calculation_method (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null*)

Gets current calculation method of SFW deals

The method retrieves the index number which corresponds to the calculation method of a deal.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.

Return values

>0	current method index
-1	Error - Deal not opened
-2	Error - Other error
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*    void* ptid = NULL;
*    //deal has been opened
*
*    int ret = get_calculation_method(ptid, null);
*    if(ret <= 0)
*    {
*        //error handling
*    }
*
```

Note

If successful returned, the value is one of PREPAY_DEFAULT_CALC_METHOD_TYPE.

22.3.3.32 int CHASAPI get_cdo_date_info (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* CDO_DATE_INFO ∗ *date_info*)

Retrieves date information of current deal.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	NULL for parent deal or name of the underlying deal.
out	*date_info*	A pointer to a structure CDO_DATE_INFO which contains the date information of a CDOnet deal.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid pointer to structure CDO_DATE_INFO.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     CDO_DATE_INFO date_info;
*     int ret = get_cdo_date_info(pDeal, NULL, &date_info);
*
```

22.3.3.33 int CHASAPI get_cdo_test_flow (void ∗ *tid,* TEST_TYPE *test_type,* const char ∗ *test_name,* CDO_TEST_FLOW ∗ *flow_test*)

Retreives test projection for current deal.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*test_type*	Test type.
in	*test_name*	The name the test.
out	*flow_test*	The test projection result.

Return values

0	Success.
-1	Error - Deal not opened and run.
-2	Error - Invalid test type.
-3	Error - Specified test not found.
-3	Error - Invalid pointer for flow_test.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open and run
*
*     CDO_TEST_FLOW testflow;
*     int ret = get_cdo_test_flow(pDeal, TEST_IC, "A1", &testflow);
*
```

22.3.3.34 int CHASAPI get_cdo_test_info (void ∗ *tid,* short ∗ *test_size,* CDO_TEST_INFO ∗ *test_info*)

Retreives test information for current deal.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*test_size*	Size of test_info vector.
out	*test_info*	A pointer to the vector of test_info.

Return values

>=0	Number of tests that have been returned
-1	Error - Deal not opened
-2	Error - Invalid input for test_size
-3	Error - Invalid pointer for test_info
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void *pDeal = NULL;
*      // deal has been opened and run
*
*      CDO_TEST_INFO testinfos[10]={0};
*      int num_test = 10;
*      int ret = get_cdo_test_info(pDeal, &num_test, &testinfos[0]);
*
```

22.3.3.35 int CHASAPI get_china_bond_info_by_tranche (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *bondid,* CHINA_BOND_INFO ∗ *bond_info*)

This method exposes get china bond info by tranche in WSAAPI

Since

3.4.1

Availability SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	Pass NULL for main deal or remic name for underlying deal.
in	*bondid*	A pointer to the name of the bond.
out	*bond_info*	A pointer to the returned china bond info.

Return values

0	No error.
-1	Error - Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "HEXIANG171");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*        const char* bondid = {"A1"};
*        CHINA_BOND_INFO bond_info;
*        assert(0, get_china_bond_info_by_tranche(pDeal, NULL, bondid, &
       bond_info));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.36 int∗ CHASAPI get_coll_cf_dates (void ∗ *tid*)

This method returns the cash flow dates of the collateral.

Since

2.1.0

Availability CDOnet, SFW

Precondition

deal is run

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

>0	A pointer to an array which stores the dates requested. The size of the vector is MAX_PERIODS.
NULL	Cashflow is not available. Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*    assert(0 == open_deal_ex(pDeal, pCmo));
*
*    assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*
*    // add settings for simulation and run simulation
*
*    assert(NULL != get_bond_cf_dates(pDeal, "A2")); // call after run simulation
*    assert(NULL != get_coll_cf_dates(pDeal));       // call after run simulation
*
*    assert(0 == close_deal_ex(pDeal, pCmo));
*    delete pCmo;
*    pCmo = NULL;
*
```

22.3.3.37 int CHASAPI get_coll_cf_length (void ∗ *tid*, short *path*)

This function returns the length of available cash flows of collateral in a specified run.

Since

2.1.0

Availability CDOnet, SFW

Precondition

open_deal_ex() has been called.
run_monte_carlo_simulation() or run_default_probability_distribution() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*path*	The index of the path whose collateral cashflows are being requested, 0 for the average collateral cashflows.

Return values

>=0	Success. Return the length of available collateral cashflows in the specified run.
-1	Deal not opened.
-3	Error - Current simulation engine is not set to any available one.
-4	Error - Invalid path.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
    *    void* pDeal = NULL;
    *    CMO_STRUCT *pCmo = new CMO_STRUCT();
    *    memset(pCmo, 0, sizeof(*pCmo));
    *    strcpy(pCmo->dealid, "ACE06NC1");
    *
    *    set_engine_preference(
        PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
    *    assert(0 == open_deal_ex(pDeal, pCmo));
    *
    *    assert(0 == set_simulation_engine(pDeal,
        SIMULATION_MONTE_CARLO));
    *    assert(0 == run_monte_carlo_simulation(pDeal));
    *
    *    assert(get_coll_cf_length(pDeal, 1) > 0);
    *
    *    assert(0 == close_deal_ex(pDeal, pCmo));
    *    delete pCmo;
    *    pCmo = NULL;
    *
```

22.3.3.38 double∗ CHASAPI get_collateral_flow_sim (void ∗ *tid*, short *path*, int *flow_identifier*)

This method returns the simulation collateral cashflow calculated. The maximum for path number of cash flows populated is 100 by default. To remove the limit, please call function remove_simulation_cashflow_populated_limit.

Since

2.1.0

Availability CDOnet, SFW

Precondition

run_monte_carlo_simulation() or run_default_probability_distribution() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*path*	The path number of the requested collateral cashflow.0 for the average collateral flow.
in	*flow_identifier*	The collateral cashflow identifier being requested.

Return values

OTHER	Pointer to the vector of cashflows.
NULL	Error - Call get_deal_error_msg().

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*       MONTE_CARLO_ASSUMPTION basic_assumption;
*       // assign members of basic_assumption
*       MONTE_CARLO_DEF_PPY_REC_ASSUMPTION def_ppy_rec_assumption;
*       // assign members of def_ppy_rec_assumption
*       assert(0 == set_monte_carlo_assumption(pDeal, &basic_assumption, &
      def_ppy_rec_assumption));
*       assert(0 == run_monte_carlo_simulation(pDeal));
*
*       double* losses = get_collateral_flow_sim(pDeal, 1,
      FLOW_COLLATERAL_LOSSES);
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.39 int CHASAPI get_coupon_stepup_date (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, char ∗ *date*)

This method returns the coupon step-up date of a deal.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*date*	A pointer to a null-terminated string (of format MM/DD/YY). This parameter must be pre-allocated with at least 11 characters.

Return values

0	The issued date of the requested pool group has been obtained successfully.
-1	Error - Deal not opened.
-2	Error - Invalid pointer to date.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT;
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "PARAGONM11");
*
*      set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(tid, pCmo));
*
*      char couponStepupDate[11] =  {0};
*      assert(0 == get_coupon_stepup_date(pDeal, NULL, couponStepupDate));
*
*      assert(0 == run_deal_ex(tid, pCmo));
*      assert(0 == close_deal_ex(tid, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.40 int CHASAPI get_currencies (void ∗ *tid,* char ∗ *currencies[]*)

Retrieves currencies used in the deal.

New feature Subject to change

Since

2.0.0

Availability SFW, CDOnet, CHS

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*currencies*	a pointer to the currency index. If it's NULL, the number of currencies used in the deal will be returned. If it's not NULL, the pointers will be updated so that they points to the addresses where store the names of the currencies used in the deal and the number of currencies used will be returned by the method.

Return values

>=0	Number of currencies used in the deal
-1	Error - Deal not opened
-2	Error - Invalid currencies pointer
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     int curr_num = get_currencies(pDeal, NULL);
*     std::vector<char> curr_buf(curr_num*4);
*     std::vector<char*> currencies(curr_num);
*     for(int i = 0; i<curr_num; i++ )
*     {
*         currencies[i] = &curr_buf[i*4];
*     }
*     int ret = get_currencies(pDeal, &currencies.front());
*     if (ret < 0)
*     {
*         // error handle
*     }
*
```

22.3.3.41 ENGINE_TYPE CHASAPI get_current_deal_engine (void * *tid*)

This is a convenience method provided in the wrapper. It can be called after the deal is opened to retrieve the engine that is being used for processing the currently open deal. It will return UNKNOWN_ENGINE if no deal is currently open.

Since

0.9.0

Availability ALL

Parameters

in	*tid*	The deal/scenario object identifier. Null if using nonthread safe calls.

Return values

UNKNOWN_ENGINE	Unknown engine (no deal open yet).
CHS_ENGINE	Current deal opened by CHS engine.
SFW_ENGINE	Current deal opened by SFW engine.
CDONET_ENGINE	Current deal opened by CDOnet engine.

Note

Call this function after opening a deal to retrieve the library it's from. Used for real-time processing on the current open deal. It will return UNKNOWN_ENGINE if no deal is currently open. See enums of ENGINE_T-YPE for the valid values this function returns.

Example:

```
*      void* tid=NULL;
*      CMO_STRUCT cmo={};
*      strcpy(cmo.dealid, "AAM0401");
*
*      set_engine_preference(PICK_CHS_ENGINE_FORMAPPED_DEALS);
*      open_deal_ex(tid, &cmo);
*
*      // The expected engine type for Deal AAM0401 should be CHS_ENGINE.
*      int engine_type = get_current_deal_engine(tid);
*
*      close_deal_ex(tid, &cmo);
*
```

22.3.3.42 int CHASAPI get_current_edf_scenario (void ∗ *tid*)

This function will return the index of the SEDF credit scenario that is using under current opened deal.

Since

2.0.1

Availability CDOnet

Precondition

open_deal_ex() has been called.
The current credit model has been set to SEDF with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

>0	Index of scenario which applying current deal .
-1	Error - Deal not open.
-3	Error - Current credit model is not SEDF.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "1776");
*
*      set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_SEDF_SETTINGS, false));
*
*      int scenario_count = get_edf_scenarios(pDeal, NULL);
*      assert(scenario_count > 0);
*
*      assert(0 == set_current_edf_scenario(pDeal, scenario_count-1));
*
*      assert(scenario_count-1 == get_current_edf_scenario(pDeal));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

- get_edf_scenarios()
- set_current_edf_scenario()

22.3.3.43 int CHASAPI get_current_moodys_cmm_scenario (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, char ∗ *cmm_scenario*)

Gets current CMM scenario of SFW deal.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*cmm_scenario*	A pointer to the user allocated string to store the returned CMM scenario, it need at least 20 characters for the scenario name.

Return values

0	Success.
-3	Error - Current credit model is not CMM.
-99	Error - Call get_deal_error_msg() for detail.

See Also

- get_moodys_cmm_scenarios()
- set_current_moodys_cmm_scenario()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "CMBS_BOA00002");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       open_deal_ex(tid, pCmo);
*
*       int scenario_count = get_moodys_cmm_scenarios(tid, NULL, NULL);
*       assert(scenario_count > 0);
*
*       char **scenarios = new char*[scenario_count];
*       for (int i = 0; i < scenario_count; ++i)
*        scenarios[i] = new char[20];
*       assert(scenario_count == get_moodys_cmm_scenarios(tid, NULL, scenarios));
*
*       set_current_moodys_cmm_scenario(tid, NULL, scenarios[3]);
*
*       char current_scenario[20];
*       get_current_moodys_cmm_scenario(tid, NULL, current_scenario);
*       assert(0 == strcmp(scenarios[3], current_scenario));
*
*       run_deal_ex(tid, pCmo);
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.44 int CHASAPI get_current_mpa_scenario (void ∗ *tid*)

This function will return the index of the MPA credit scenario that is using under current opened deal.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

>0	Index of scenario which applying current deal .
-1	Error - Deal not open.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*
*       int scenario_count = get_mpa_scenarios(pDeal, NULL);
*       assert(scenario_count > 0);
*
*       assert(0 == set_current_mpa_scenario(pDeal, scenario_count-1));
*
*       assert(scenario_count-1 == get_current_mpa_scenario(pDeal));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

- get_mpa_scenarios()
- set_current_mpa_scenario()

22.3.3.45 int CHASAPI get_current_pa_scenario (void ∗ *tid*)

This function will return the index of the PA credit scenario that is using under current opened deal.

Since

2.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

>0	Index of scenario which applying current deal .
-1	Error - Deal not opened.
-3	Error - PA model is not setup.
-99	Error - Call get_deal_error_msg() for detail.

See Also

- get_pa_scenarios()
- set_current_pa_scenario()

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
     MOODYS_PA_SETTINGS, false));
*      assert(0 == set_current_pa_scenario(pDeal, 1));
*
*      int cur_scen = get_current_pa_scenario(pDeal);
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.46 int CHASAPI get_custom_call_status (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* BOOLYAN ∗ *status*)

This method returns the status of the "Custom Call" of a deal.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*status*	TRUE/FALSE indicate custom call checkbox value of a deal.

Return values

0	No error.
-1	Error - Deal not opened.
-2	Error - Invalid pointer to status.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "PARAGONM11");
*
*       set_engine_preference(
        PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       BOOLYAN isCustomCall = false;
*       assert(0 == get_custom_call_status(tid, NULL, &isCustomCall));
*
*       assert(0 == run_deal_ex(tid, pCmo));
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.47 int CHASAPI get_deal_account_avail (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, char ∗ *account_names[]*, DEAL_ACCOUNT_INFO *account_info[]*, unsigned int *account_size*)

Retrieves the name and account info of the available accounts in current deal.

This method gets all the deal accounts info, if the deal accounts are available.

Since

1.1.0

Availability CDOnet, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*account_size*	Fields size of account_names and account_info.
out	*account_names*	A pointer to a client-allocated array of characters strings to store names of the account. At least 11 characters should be allocated for each string.
out	*account_info*	Fields populated with account information.

Returns

Number of accounts in deal.

Return values

>=0	Number of accounts in deal
-1	Deal not open
-2	Invalid account_names or account_info

Example:

```
*      void * pDeal = NULL;
*      // Deal is already open.
*
*      int account_num = get_deal_account_avail(pDeal, NULL, NULL, NULL, 0);
*      std::vector<char> name_buf(account_num*11);
*      std::vector<char*> names(account_num);
*      for(int i = 0; i < account_num; ++i)
*      {
*          names[i] = &name_buf[i*11];
*      }
*      std::vector<DEAL_ACCOUNT_INFO> info(account_num);
*      assert(account_num == get_deal_account_avail(pDeal, NULL, &names.front(), NULL
      , account_num));
*      assert(account_num == get_deal_account_avail(pDeal, NULL, NULL, &info.front(),
       account_num));
*      assert(account_num == get_deal_account_avail(pDeal, NULL, &names.front(), &
      info.front(), account_num));
*
```

22.3.3.48 int CHASAPI get_deal_account_flow (void * *tid*, const char * *reremic_deal_id_or_null*, char * *account_name*, MOODYS_ACCOUNT_CASHFLOW * *cf*)

Retrieves account flows for running dynamic cashflows.

New feature Subject to change

Since

4.0.0

Availability CDOnet, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*loan_number*	The 1-based index of the loan.
in	*account_name*	The account name, refer to the field id from DEAL_ACCOUNT_INFO.
in	*cf*	The account level cash flow data.

Return values

NULL	Error - Call get_deal_error_msg().
OTHER	Pointer to the vector of cashflows.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF00001");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*      assert(0 == run_deal_ex(pDeal, pCmo));
*      MOODYS_ACCOUNT_CASHFLOW cf;
*      memset(&cf, 0, sizeof(MOODYS_ACCOUNT_CASHFLOW));
*      int ret = get_deal_account_flow(pDeal, "xxx", &cf);
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.49 int CHASAPI get_deal_fee (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, MOODYS_FEE_STRUCT *fee_info[ ]*, int *size*, int ∗ *num_fees*)

This method returns a list of fee information(name, type, value, day count info) of a deal.

Since

2.9.0

Availability SFW, CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*fee_info*	A client-allocated array of MOODYS_FEE_STRUCT for the fee information which will be stored
in	*size*	The length of the array fee_info that the user has passed down.
out	*num_fees*	Total number of available Fees.

Return values

>=0	Actual number of fees returned.
-1	Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*     void *pDeal = NULL;
*     //deal has been opened
*
*     MOODYS_FEE_STRUCT fee_info[1] = {};
*     memset(fee_info, 0, sizeof(MOODYS_FEE_STRUCT)*1);
*     int num_fees = 0;
*     int iret = get_deal_fee(pDeal, NULL, fee_info, 1, &num_fees);
*     if(iret > 0 && num_fees > iret)
*     {
*         std::vector<MOODYS_FEE_STRUCT> fee_info_vec(num_fees);
*         iret = get_deal_fee(pDeal, NULL, &fee_info_vec.front(), num_fees, &num_fees);
*     }
*
```

Note

Pass NULL for fee_info to get the number of fees. The fees can be overridden by calling set_deal_fee_override before calling run_deal_ex().

22.3.3.50 double∗ CHASAPI get_deal_fee_flow (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* char ∗ *fee_name*)

This method returns the fee flow of a specific fee item through fee name of a deal.

Since

3.3.0

Availability SFW, CDOnet

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*fee_name*	The fee name, can be retrieved from function get_deal_fee.

Return values

NULL	Error - Call get_deal_error_msg().
OTHER	Pointer to the vector of cashflows.

Example:

```
*       void *pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "STO16IIBV");
*
*       set_engine_preference(
        PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       MOODYS_FEE_STRUCT fee_info[1] = {};
*       memset(fee_info, 0, sizeof(MOODYS_FEE_STRUCT)*1);
*       int num_fees = 0;
*       int iret = get_deal_fee(pDeal, NULL, fee_info, 1, &num_fees);
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       double * flow = get_deal_fee_flow(pDeal, NULL, feeinfo[0].fee_name);
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.51 int CHASAPI get_deal_hedge (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* MOODYS_HEDGE_STRUCT *hedge_info[ ],* int *size,* int ∗ *num_hedges*)

This method returns a list of hedge information(ids, descriptions, margin, hedge override info) of an SFW deal.

Since

2.9.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*hedge_info*	A client-allocated array of MOODYS_HEDGE_STRUCT in hedges informa-tionwhich will be stored
in	*size*	The length of the array hedge_info that the user has passed down.
out	*num_hedges*	Total number of available hedges.

Return values

>=0	Actual number of hedges returned.
-1	Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*     void *pDeal = NULL;
*     //deal has been opened
*
*     MOODYS_HEDGE_STRUCT hedgeinfo[1] = {};
*     memset(hedgeinfo, 0, sizeof(MOODYS_HEDGE_STRUCT)*1);
*     int num_hedges = 0;
*     int iret = get_deal_hedge(pDeal, NULL, hedgeinfo, 1, &num_hedges);
*     if(iret > 0 && num_hedges > iret)
*     {
*         std::vector<MOODYS_HEDGE_STRUCT> hedgeinfovec(num_hedges);
*         iret = get_deal_hedge(pDeal, NULL, &hedgeinfovec.front(), num_hedges, &num_hedges);
*     }
*
```

Note

Pass NULL for hedge_info to get the number of hedges. The hedges can be overridden by calling set_deal_-hedge_override before calling run_deal_ex().

22.3.3.52 long CHASAPI get_deal_info_ex (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, MOODYS_DEAL_INFO ∗ *deal_info*)

Retrieves additional deal information

New feature Subject to change

Since

2.6.4

Availability CDOnet, SFW, CHS

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	Pass NULL for main deal or remic name for underlying deal
out	*deal_info*	additional deal information, .

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     //deal has been opened
*
*     MOODYS_DEAL_INFO mdi={};
*     int ret = get_deal_info_ex(pDeal, NULL, &mdi);
*     if (0 != ret)
*     {
*         // error handle
*     }
*
```

22.3.3.53 int CHASAPI get_deal_refinance_date (void ∗ *tid,* int *refinance_dates_array[ ],* int *num_dates*)

This method returns the refinance dates of a deal.

Since

2.7.0

Availability CDONET

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*refinance_dates-_array*	A pointer to the array of deal reprice/refinance dates (YYYYMMDD).
in	*num_dates*	Fields size of refinance_dates_array.

Return values

>=0	Number of refinance dates.
-1	Error - Deal not opened.
-2	Error - Invalid refinance_dates_array and num_dates passed.
-99	Error - Other error, call get_deal_error_msg() for detail.

Note

refinance_dates_array required to be null and num_dates required to be zero when the first call

Example:

```
*     void* tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT;
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "EURO3");
*
*     set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*     assert(0 == open_deal_ex(tid, pCmo));
*
*     int num_dates;
*     num_dates = get_deal_refinance_date(pDeal, NULL, 0);
*
*     int refinance_dates_array[] = new int[num_dates];
*     assert(num_dates == get_deal_refinance_date(pDeal, refinance_dates_array,
      num_dates));
*
*     assert(0 == close_deal_ex(tid, pCmo));
*     delete pCmo;
*     pCmo = NULL;
*
```

22.3.3.54 int CHASAPI get_deal_update_id (void ∗ *tid*, char ∗const *update_id*, const int *len*)

Retrieves deal update ID.

Since

1.4.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*update_id*	A pointer to a client-allocated character array.
in	*len*	The size of update_id.

Return values

0	No error
-99	Invalid dso identifier (tid) or other errors, for details call get_deal_error_msg()

Example:

```
*      char updateid[20]={0};
*      void *tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AL2010-A");
*      strncpy(pCmo->settlement_date,"05/01/11",strlen("05/01/11"));
*
*      open_deal_ex(tid,pCmo);
*
*      get_deal_update_id(tid,updateid, 20);
*
*      close_deal_ex(tid,pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.55 double∗ CHASAPI get_dpd_current_default_timing (void ∗ *tid*)

This methods returns the default timing curves for current assets. If the user input for the default timing does not add up to 100%, the simulator will scale them up/down to 100% when running the simulation. Users can use this method to get the actual values that are used after running the simulation.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

>0	A pointer to an array which stores the default timing for current assets. The size of the array will be MAX_PERIODS.
NULL	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*       // add settings for DPD
*
*       assert(0 == run_default_probability_distribution(pDeal));
*
*       double *currDefaultTimingCurve = get_dpd_current_default_timing(pDeal)
      ;
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.56 int CHASAPI get_dpd_el_pd_factors (void ∗ *tid*, double ∗ *el_factor*, double ∗ *pd_factor*)

This method returns the E.L. and P.D. table factors.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*el_factor*	E.L. factor.
out	*pd_factor*	D.P. factor.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF00001");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
     SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*      double elFactor = 0.0, pdFactor = 0.0;
*      assert(0 == get_dpd_el_pd_factors(pDeal, &elFactor, &pdFactor));    // can call
       it before/after run.
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.57 int CHASAPI get_dpd_results (void ∗ *tid,* const char ∗ *bondid,* DPD_RESULT ∗ *result*)

This method returns the Default Probability Distribution Simulation results.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
run_default_probability_distribution() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The name of the tranche whose results are being requested. The length should be 20.
out	*result*	A user allocated structure in which the results of the Default Probability Distribution will be stored.

Return values

>=0	number of bonds whose Default Probability Distribution Simulation results have been populated.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*       // add settings for DPD
*
*       assert(0 == run_default_probability_distribution(pDeal));
*
*       DPD_RESULT result;
*       assert(0 == get_dpd_results(pDeal, "A1", &result));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.58 double* CHASAPI get_dpd_revolving_default_timing (void * *tid*)

This methods returns the default timing curves for revolving assets. If the user input for the default timing does not add up to 100%, the simulator will scale them up/down to 100% when running the simulation. Users can use this method to get the actual values that are used after running the simulation.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

>0	A pointer to an array which stores the revolving default timing for current assets. The size of the array will be MAX_PERIODS.
NULL	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF00001");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
     SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*      // add settings for DPD
*
*      assert(0 == run_default_probability_distribution(pDeal));
*
*      double *revDefaultTimingCurve = get_dpd_revolving_default_timing(
     pDeal);
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.59 int CHASAPI get_dpd_scenarios (void * *tid,* DPD_SCENARIO * *scenarios,* short *size_scenarios*)

This method will return an array of structures which contains the user defined or system generated scenarios. If NULL is passed down for parameter scenarios, then the method will return the size that is needed to store all the scenarios.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
run_default_probability_distribution() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
`out`	*scenarios*	a user allocated array which stores the information of default/loss scenarios. Call get_dpd_scenarios(tid, NULL, 0) to get the count of scenarios.
`in`	*size_scenarios*	the size of scenarios. Call get_dpd_scenarios(tid, NULL, 0) to get the count of scenarios.

Return values

>=0	Number of scenarios that have been populated. Number of scenarios that have been populated. Size that is needed to store all the scenarios if NULL is passed down for parameter scenarios.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       // settings for DPD
*
*       assert(0 == run_default_probability_distribution(pDeal));
*
*       int scenariosNum = get_dpd_scenarios(pDeal, 0, 0);
*       assert(4 == scenariosNum); // assume scenario number is 4
*
*       DPD_SCENARIO scenarios[4];
*       assert(4 == get_dpd_scenarios(pDeal, scenarios, sizeof(scenarios)/sizeof(scenarios[
      0])));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.60 int CHASAPI get_dpd_threshold (void * *tid*, const char * *rating*, short *year*, double * *threshold*)

This method returns the rating threshold.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*rating*	The rating whose corresponding threshold will be updated.

in	*year*	The year of the rating of which the corresponding threshold will be updated.
out	*threshold*	The threshold value of the specified rating of the specified year.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF00001");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
     SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*      double threshold = 0.0;
*      assert(0 == get_dpd_threshold(pDeal, "Aaa", 3, &threshod)); // can call it
       before/after run.
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.61 int CHASAPI get_edf_scenarios (void ∗ *tid,* char ∗ *scenario_list[ ]*)

This function will get SEDF model available scenarios.

Since

2.0.1

Availability CDOnet

Precondition

open_deal_ex() has been called.
The current credit model has been set to SEDF with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*scenario_list*	The list of scenarios. If this parameter is NULL, this function will return with number of available scenarios. The memory of scenario_list should be allocated by user. The lengh of each element of scenario_list should be 20.

Return values

>=0	Success. Number of available scenarios.
-1	Deal not opened.
-2	Error - Invalid pointer.
-3	Error - SEDF model is not setup.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      int buflen = 50;
*      char value[buflen]={0};
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "1776");
*
*      set_engine_preference(
     PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
     MOODYS_SEDF_SETTINGS, false));
*
*      int scenario_count = get_edf_scenarios(pDeal, NULL);
*      assert(scenario_count > 0);
*
*      std::vector<char *> scenario_list(scenario_count);
*      std::vector<char> scenario_strs(scenario_count*20);
*
*      for (int i = 0; i < scenario_count; ++i)
*          scenario_list[i] = &scenario_strs[i*20];
*      assert(scenario_count == get_edf_scenarios(pDeal, &scenario_list[0]));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.62 int CHASAPI get_exchange_rate (void ∗ *tid,* const char ∗ *currency,* double ∗ *pval*)

Get specified currency exchange rate per global currency.

New feature Subject to change

Since

2.0.0

Availability SFW CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The specified currency name.
out	*pval*	Exchange rate value pointer.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid currency name
-3	Error - Invalid exchange rate value pointer
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     double rate = 0.0;
*     int ret = get_exchange_rate(pDeal, "USD", &rate);
*     if (0 != ret)
*     {
*         // error handle
*     }
*
```

22.3.3.63 int CHASAPI get_first_principal_pay_month (void ∗ *tid,* const char ∗ *bondid,* char ∗ *first_prin_pay_month*)

This method returns the first principal payment month corresponding to the requested bond.

Since

4.0.0

Availability CDOnet, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The name of the bond whose date is being requested.
out	*first_prin_pay_-month*	The time when the tranche receives its first principal. A pointer to a null-terminated string (YYYYMM). This must be pre-allocated with at least 9 characters.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid pointer.
-3	Error - Bond not found.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg().

Example:

```
*     void* tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT();
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "DRYDEN34");
*
```

```
*     assert(0 == open_deal_ex(tid, pCmo));
*     assert(0 == run_deal_ex(tid,pCmo));
*
*     const char* bondid = "1A1";
*      char first_prin_pay_month[9];
*     int nRet = get_first_principal_pay_month(tid, bondid,
      first_prin_pay_month);
*     if(nRet !=0)
*     {
*          //error handle;
*     }
*
*      assert(0 == close_deal_ex(tid, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.64 double∗ CHASAPI get_forward_interest_rates (void ∗ *tid,* const char ∗ *currency,* short ∗ *rate_type*)

Get the forward interest rate curves after generate_forward_interest_rates() is successfully called.

Since

3.0.0

Availability SFW, CHS, CDONET

Precondition

generate_forward_interest_rates() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	ISO currency code (e.g., USD).
in	*rate_type*	Type of interest rate. ∗rate_type must be one of: enums of INDEX_TYPE (In indextypes.h). enums of INDEX_TYPE_EX (SFW and CDOnet deals).

Return values

NULL	Forward curve for the requested interest rate is not available.
Other	A pointer to an array which stores the forward curve for the requested interest rate. The array length is MAX_PERIODS(=612).

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "CQSCLO2");
*
*      open_deal_ex(tid, pCmo);
*
*      short idx = LIBOR_1;
*      double rate = 0.00853;
*      set_index_rate(pDeal, "USD", &idx, 0, &rate);
*      generate_forward_interest_rates(tid);
*
*      double *pRate = get_forward_interest_rates(tid, "USD", &idx);
*
*      assert(0 == close_deal_ex(tid, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.65 int CHASAPI get_global_currency (void ∗ *tid,* char ∗ *currency_index*)

Retrieves global currency of deal.

New feature Subject to change

Since

2.0.0

Availability SFW, CDOnet, CHS

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*currency_index*	a pointer to the global currency index.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void *pDeal = NULL;
*      //deal has been opened
*
*      char global_currency[4];
*      int ret = get_global_currency(pDeal, global_currency);
*      if (0 != ret)
*      {
*          // error handle
*      }
*
```

22.3.3.66 int CHASAPI get_global_reinvestment (void ∗ *tid,* GLOBAL_REINVESTMENT_INFO ∗ *reinv_info,* short *pool_size,* GLOBAL_REINVESTMENT_ASSET_INFO *pool_info[ ]*)

Gets the global reinvestment setting information.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*reinv_info*	A pointer to the global reinvestment info, refer to GLOBAL_REINVESTMENT-_INFO.
in	*pool_size*	The length of the vector of global reinvestment portfolio.
out	*pool_info*	A pointer to a vector containing the global reinvestment portfolio, refer to GL-OBAL_REINVESTMENT_ASSET_INFO.

Return values

>=	Actual number of global reinv assets
-1	Error - Deal not opened
-2	Error - Invalid input for pool_size
-3	Error - Invalid pointer for pool_info
-4	Error - invalid pointer for reinv_info
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     GLOBAL_REINVESTMENT_INFO reinv_info;
*     const int nAssets = 5;
*     std::vector<GLOBAL_REINVESTMENT_ASSET_INFO> vecAssets(nAssets);
*     int actual_num = get_global_reinvestment(pDeal, reinv_info, nAssets, &
      vecAssets.front());
*     if (actual_num > nAssets)
*     {
*         vecAssets.clear();
*         vecAssets.resize(actual_num);
*         get_global_reinvestment(pDeal, reinv_info, actual_num, &vecAssets.front())
      ;
*     }
*
```

22.3.3.67 int CHASAPI get_group_issue_date (void ∗ *tid,* int *group_number,* char ∗ *date*)

This method gets the issue date of a collateral group by group number.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*group_number*	The 1-based group number of the reqeusted pool group.
out	*date*	A pointer to a null-terminated string (of format YYYYMMDD). This parameter must be pre-allocated with at least 11 characters.

Return values

0	The issued date of the requested pool group has been obtained successfully.
-1	Error - Deal not opened.
-2	Error - Invalid pool group number.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT;
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "CMBS_BOA06006");
*
*      set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(tid, pCmo));
*
*      int group_number = 1;
*      char date[11];
*
*      assert(0 == get_group_issue_date(tid, group_number, date));
*      assert(0 == strcmp("20061101", date));
*
*      assert(0 == run_deal_ex(tid, pCmo));
*      assert(0 == close_deal_ex(tid, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.68 double∗ CHASAPI get_haircut_flow (void ∗ *tid*, CDO_HAIRCUT_TYPE *haircut_type*)

Retreives haircut projection for current deal.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*haircut_type*	Haircut type.

Return values

NULL	Invalide haircut type.
Other	A pointer to the specified haircut vector, the length of vector is MAX_PERIODS

Example:

```
*      void *pDeal = NULL;
*      // deal has been opened and run
*
*      double *ret = get_haircut_flow(pDeal, HAIRCUT_1);
*
```

22.3.3.69 double∗ CHASAPI get_index_rate (void ∗ *tid,* const char ∗ *currency,* short ∗ *idx*)

Gets the rates for the given index and currency.

New feature Subject to change

Since

2.0.0

Availability ALL

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The specified currency name.
in	*idx*	the rate index type. ∗idx must be one of: • enums of INDEX_TYPE (In indextypes.h). • enums of INDEX_TYPE_EX (SFW and CDOnet deals).

Return values

NULL	The requested index rate was not retrieved successfully.
Other	A pointer to an array which stores the rates used for the index rate,the array length is MAX_PERIODS(=612).

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     short index = LIBOR_3;
*     double* pRate = get_index_rate(pDeal, "USD", &index)
*     if (NULL == pRate)
*     {
*         // error handle
*     }
*
```

Note

For CHS engine, the currency just support the "USD".

22.3.3.70 long CHASAPI get_indices (void ∗ *tid,* const char ∗ *currency,* short ∗ *ps_rates*)

Returns number of market rate indices per specified currency used by currently open deal (includes bonds, collateral, hedges, accounts, and waterfall script) and populates ps_rates vector to indicate which market rate indices are used.

New feature Subject to change

Since

2.0.0

Availability ALL

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The specified currency code. It should be the alphabetic code from ISO 4217, e.g. "EUR", "USD". Please refer to `current currency & funds code list` for all the available currency codes.
out	*ps_rates*	Returns number of market rate indices used by currently open deal (includes bonds, collateral, hedges, accounts, and waterfall script) and populates ps_-rates vector to indicate which market rate indices are used.

Return values

>=0	return The number of rates used.
-1	Error - Deal not open
-2	Error - Invalid currency name
-3	Error - Invalid ps_rates pointer

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     short idx_rates[MAX_INDEX_TYPES] = {0};
*     int ret = get_indices(pDeal, "EUR", idx_rates);
*     if (ret < 0)
*     {
*         // error handle
*     }
*
```

Note

For CHS engine, the currency just support the "USD".

22.3.3.71 int CHASAPI get_last_principal_pay_month (void * *tid,* const char * *bondid,* char * *last_prin_pay_month*)

This method returns the last principal payment month corresponding to the requested bond.

Since

4.0.0

Availability CDOnet, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The name of the bond whose date is being requested.
out	*last_prin_pay_-month*	The time when the tranche receives its last principal. A pointer to a null-terminated string (YYYYMM). This must be pre-allocated with at least 9 characters.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid pointer.
-3	Error - Bond not found.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg().

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "DRYDEN34");
*
*      assert(0 == open_deal_ex(tid, pCmo));
*      assert(0 == run_deal_ex(tid,pCmo));
*
*      const char* bondid = "1A1";
*       char last_prin_pay_month[9];
*      int nRet = get_last_principal_pay_month(tid, bondid, last_prin_pay_month)
       ;
*      if(nRet !=0)
*      {
*           //error handle;
*      }
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.72 int∗ CHASAPI get_loan_dates (void ∗ *tid,* int *loan_number*)

Retrieves loan pay dates

New feature Subject to change

Since

2.9.0

Availability SFW

Precondition

run_deal_ex() has been called. calculation level set to CALC_LEVEL_FULL_WITH_LOAN by set_deal_calc-_level

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan.

Return values

>0	A pointer to an array which stores the loan pay dates. The size of the vector is MAX_PERIODS.
NULL	pay dates is not available. Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(ptid, CALC_LEVEL_FULL_WITH_LOAN, 1);

*       run_deal_ex(pDeal, pCmo);
*       // Deal is already run successfully.
*
*       // Get loan dates.
*       assert(NULL != get_loan_dates(pDeal, 1));
*
*       close_deal_ex(pDeal, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.73 int CHASAPI get_loan_edf (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* long *loan_num,* double *pd[ ],* int *length*)

This method retrieves EDF data of a specified loan.

Since

2.0.1

Availability CDOnet, SFW

Precondition

open_deal_ex() has been called.
For CDONet deal, the current credit model should been set to SEDF with API set_moodys_credit_model_-settings(), but SFW deal does not need.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*loan_num*	Ordinal collateral number in a deal.
out	*pd*	A pointer to a client-allocated array of double to store EDF data. At least 5 elements should be allocated.
in	*length*	The number of the elements pointed by pd.

Return values

0	Success.
-1	Deal not open.
-2	Invalid loan_num or tid.
-3	Current credit model is not Stress EDF.
-4	Invalid EDF data array.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "1776");
*
*       set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       double pd[5]={0.0};
*       assert(0 == get_loan_edf(pDeal, NULL, 2, pd, 5));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.74 double∗ CHASAPI get_loan_flow (void ∗ *tid*, int *loan_number*, const char ∗ *reremic_deal_id_or_null*, int *flow_identifier*)

Retrieves loan level cashflows

New feature Subject to change

Since

2.7.0

Availability SFW

Precondition

run_deal_ex() has been called. calculation level set to CALC_LEVEL_FULL_WITH_LOAN by set_deal_calc-_level

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan.
in	*reremic_deal_id-_or_null*	Pass NULL for main deal or remic name for underlying deal
in	*flow_identifier*	Cashflow identifier. Currently loan level cashflow support: FLOW_COLLATERAL_BALANCE FLOW_COLLATERAL_INTEREST FLOW_COLLATERAL-_PRINCIPAL FLOW_COLLATERAL_CASH

Return values

NULL	The requested loan cashflow was not retrieved successfully.
Other	A pointer to an array which stores the requested loan cashflow,the array length is MAX_PERIODS.

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT cmo;
*       memset(&cmo,0,sizeof(CMO_STRUCT));
*       strcpy(cmo.dealid,"ACE06NC1");
*
*       open_deal_ex(ptid,&cmo);
*
*       set_deal_calc_level(ptid, CALC_LEVEL_FULL_WITH_LOAN, 1);

*       run_deal_ex(ptid,&cmo);
*       // Deal is already opened successfully.
*       // Deal is already run successfully.
*
*       // Get bond flow of balance.
*       double* pLoanBalance = get_loan_flow(ptid, 1, NULL,
      FLOW_COLLATERAL_BALANCE);
*
*       close_deal_ex(ptid, &cmo);
*
```

22.3.3.75 int CHASAPI get_loan_flow_ex (void ∗ *tid,* int *loan_number,* MOODYS_LOAN_CASHFLOW ∗ *cf*)

Retrieves brief loan cashflow to the cashflow structure MOODYS_LOAN_CASHFLOW

New feature Subject to change

Since

3.3.0

Availability CDOnet, SFW

Precondition

run_deal_ex() has been called. calculation level set to CALC_LEVEL_FULL_WITH_LOAN by set_deal_calc-_level

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan.
in	*cf*	The loan level cash flow data.

Return values

0	No error
-1	Error - Deal not opened.
-1	Error - Invalid loan number.
-1	Error - Loan not found.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(ptid, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(ptid, CALC_LEVEL_FULL_WITH_LOAN, 1);

*       run_deal_ex(ptid, pCmo);
*       // Deal is already opened successfully.
*       // Deal is already run successfully.
*
*       MOODYS_LOAN_CASHFLOW MoodysLoanFlow;
*       memset(&MoodysLoanFlow, 0, sizeof(MOODYS_LOAN_CASHFLOW));
*       int ret = get_loan_flow_ex(ptid, 1, &MoodysLoanFlow);
*
*       close_deal_ex(ptid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.76 int CHASAPI get_loan_flow_size (void * *tid*, int *loan_number*)

Retrieves loan pay cashflow size

New feature Subject to change

Since

3.4.1

Availability CDONET, SFW

Precondition

set_whole_loan() has been called. calculation level set to CALC_LEVEL_FULL_WITH_LOAN by set_deal_-calc_level

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan.

Return values

>=0	The size of the loan cashflow.
-2	Error - Invalid loan number.
-4	Error - Other error.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(ptid, CALC_LEVEL_FULL_WITH_LOAN, 1);

*       run_deal_ex(pDeal, pCmo);
*       // Deal is already run successfully.
*
*       // Get loan cashflow size.
*       int ret = get_loan_flow_size(pDeal, 1);
```

```
*
*       close_deal_ex(pDeal, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.77 int CHASAPI get_loan_market_risk_metrics (void ∗ *tid,* int *LoanID,* METRIC_INPUT_STRUCT ∗ *metric_inputs,* METRIC_RESULTS_STRUCT ∗ *metric_results*)

This method returns the market risk metrics calculation result of a specified loan.

Since

3.2.0

Availability CDONet

Precondition

set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*LoanID*	Loan Number..
in	*metric_inputs*	The inputs for calculating metrics result.
out	*metric_results*	The pointer of the market risk result.

Return values

=0	Success
-1	Deal not open.
-2	Error - Invalid loan number.
-3	Error - Loan not found.
-4	Error - Invalid index type.
-5	Error - Invalid pointer of metric results.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*
*       METRIC_INPUT_STRUCT metric_input;
*       memset(&metric_input, 0, sizeof(METRIC_INPUT_STRUCT));
*       metric_input.clean_price = 100;
*       metric_input.apply_spread_to = APPLY_SPREAD_TO_TSY;
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       METRIC_RESULTS_STRUCT results_m;
*       memset(&results_m, 0, sizeof(METRIC_RESULTS_STRUCT));
*       assert(0 == get_loan_market_risk_metrics(pDeal, loanID, &metric_input, &
      results_m));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.78 int CHASAPI get_loan_market_risk_metrics_ex (void ∗ *tid*, int *LoanID*, METRIC_ANCHORS *anchor_type*, double *anchor_value*, APPLY_SPREAD_TYPE *apply_to*, METRIC_RESULTS_STRUCT_EX ∗ *results_ex*)

This method returns the extra market risk metrics calculation result of a specified loan.

Since

3.0.0

Availability ALL

Precondition

set_metrics_input_ex() with OAS_CAL_MODE enabled has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*LoanID*	Loan number.
in	*anchor_type*	Anchor type for metrics calculation. Available options: MARKET_PRICE or OAS, if input MARKET_PRICE then return oas result, otherwise return market price.
in	*anchor_value*	Value of the provided metric anchor, corresponding to anchor_type in METRIC_INPUT_STRUCT_EX
in	*apply_to*	The type of metric calculation apply spread to TREATURY/LIBOR Curves.
out	*results_ex*	The pointer of the market risk result.

Return values

=0	Success
-1	Deal not open.
-2	Error - Invalid loan number.
-3	Error - loan not found.
-4	Error - Invalid anchor type.
-5	Error - Invalid spread apply type.
-6	Error - Invalid pointer of metric results.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*       char *bondid = "20151A";
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       double rate_L1 = .0024563;
*       double rate_T3 = .001;
*       short index_L1 = LIBOR_1, index_T3 = TSY_3;
*       assert(0 == set_index_rate(pDeal, "USD", &index_L1, 0, &rate_L1));
*       assert(0 == set_index_rate(pDeal, "USD", &index_T3, 0, &rate_T3));
*
*       METRIC_INPUT_STRUCT_EX metric_input_ex;
*       memset(&metric_input_ex, 0, sizeof(METRIC_INPUT_STRUCT_EX));
*       metric_input_ex.shift_amt = 0.00001;
*       metric_input_ex.num_paths = 10;
*       metric_input_ex.oas_mode = ENABLE_ALL;
*       assert(0 == set_metrics_input_ex(pDeal, &metric_input_ex));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       METRIC_RESULTS_STRUCT_EX metric_result_ex;
*       memset(&metric_result_ex, 0, sizeof(metric_result_ex));
*       assert(0 == get_loan_market_risk_metrics_ex(pDeal, LoanID,
       MARKET_PRICE, 100, APPLY_SPREAD_TO_LIBOR, &metric_result_ex));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

If want to get the metric results of this method, set_metrics_input_ex must be called before.

Warning

In each path of OAS-related analysis, LIBOR 6 month spread input to MPA and PA is floored to 0.01% as required by MPA and PA; TSY 1 year and TSY 10 year input to MPA and PA is floored to -2% as required by MPA and PA.

22.3.3.79 int CHASAPI get_loan_next_reset_date (void ∗ *tid,* int *loan_number,* int ∗ *next_reset_date*)

This function gets next reset date for a given loan.

Since

3.3

Availability CDOnet

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The loan identifier.
out	*next_reset_date*	The next reset date of specified loan

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid loan number.
-3	Error - Next reset date is not available.
-99	Error - Other error, use get_deal_error_msg() to see details.

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "DRYDEN34");
*
*      assert(0 == open_deal_ex(tid, pCmo));
*      assert(0 == run_deal_ex(tid,pCmo));
*
*      int next_reset_date = 0;
*      int nRet = get_loan_next_reset_date(tid, 1, &next_reset_date);
*      if(nRet !=0)
*      {
*           //error handle;
*      }
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.80 int CHASAPI get_MA_rate_shifts_scenarios (const char * *file_path,* int *yyyymmdd,* char * *scenario_list[]*)

This function gets the available MA rate shifts scenarios list for the specified date yyyymmdd. The function looks back as many as 7 calendar days to avoid data availability gaps from weekends and holidays.

Since

3.7.0

Availability CDOnet, SFW, CHS

Parameters

in	*file_path*	Directory where the MWSA DB resides.
in	*yyyymmdd*	User input date with format YYYYMMDD. In case the MWSA DB of YYYYMM-DD is not available, look back as many as 7 days.
in	*scenario_list*	The list of scenarios. If this parameter is NULL, this function will return with number of available scenarios. The memory of scenario_list should be allocated by user. The lengh of each element of scenario_list should be 20.

Return values

>=0	Success. Number of available scenarios.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       int buflen = 50;
*       char value[buflen]={0};
*       int scenario_count = get_MA_rate_shifts_scenarios(
      get_input_path(), 20181101, NULL);
*       assert(scenario_count > 0);
*
*       std::vector<char *> scenario_list(scenario_count);
*       std::vector<char> scenario_strs(scenario_count*20);
*
*       for (int i = 0; i < scenario_count; ++i)
*           scenario_list[i] = &scenario_strs[i*20];
*       assert(scenario_count == get_MA_rate_shifts_scenarios(
      get_input_path(), 20181101, &scenario_list[0]));
*
```

22.3.3.81 int CHASAPI get_master_trigger_info (void * *tid,* const char * *reremic_deal_id_or_null,* const char * *trigger_name,* SBYTE * *breached,* char * *sub_trigger_logic,* char * *sub_trigger_names[],* char * *sub_trigger_descs[],* int *size*)

This method returns the information(breached, logic, sub-trigger names, sub-trigger descriptions) of the master trigger that the user requested.

Since

2.0.1

Availability SFW

Precondition

open_deal_ex() has been called.
run_deal_ex() has been called to get the breached.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*trigger_name*	The case-sensitive name of the trigger whose information are being requested.
out	*breached*	A pointer to an array of the projection of the status of the requested trigger. The size of the array is MAX_PERIODS.
out	*sub_trigger_-logic*	Master trigger's logic, can be either "All Yes" or "Any Yes".21 characters should be allocated for the string.
out	*sub_trigger_-names*	A pointer to a client-allocated array of character strings in which the names of the sub triggers of the requested trigger will be stored. 21 characters should be allocated for each string.
out	*sub_trigger_-descs*	A pointer to a client-allocated array of character strings in which the descriptions of the sub triggers will be stored. 1025 characters should be allocated for each string.
in	*size*	The length of the arrays that the user has passed down.

Return values

>0	Number of sub-triggers of the requested master trigger.
-1	Deal not open.
-2	Invalid pointer
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       //Deal has been run
*
*       std::vector<char> name_buf(10*21), desc_buf(10*1025);
*       std::vector<char*> names(10), descs(10);
*       for(int i = 0; i<10; i++)
*       {
*           names[i] = &name_buf[i*21];
*           descs[i] = &desc_buf[i*1025];
*       }
*       signed char status[MAX_PERIODS] = {0};
*       char sub_trigger_logic[21] = {0};
*       int ret = get_master_trigger_info(pDeal, NULL, "PRO-RATA", status,
      sub_trigger_logic, &names.front(), &descs.front(), 10);
*       if(ret < 0)
*       {
*           //Error handle
*       }
*
```

22.3.3.82 double* CHASAPI get_misc_flow (void * *tid*, int *flow_identifier*)

Returns a pointer to a vector of doubles containing deal level miscellaneous cashflow, the vector size is MAX_PERIODS.

New feature Subject to change

Since

2.0.0

Availability SFW CDOnet

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*flow_identifier*	The identifier of cashflow. Must be one of FLOW_MISC_INDENTIFIER.

Return values

NULL	The deal does not contain this cashflow field.
OTHER	Pointer to the vector of cashflows

Note

Call set_deal_calc_level() with parameter CALC_LEVEL_FULL after open_deal_ex() but before run_deal_-ex().

Example:

```
*   void*ptid=NULL;
*   CMO_STRUCT cmo;
*   memset(&cmo, 0, sizeof(CMO_STRUCT));
*   strcpy(cmo.dealid,"ACE06NC1");
*
*   open_deal_ex(ptid,&cmo);
*   set_deal_calc_level(ptid, CALC_LEVEL_FULL, 1);
*   run_deal_ex(ptid,&cmo);
*   // Deal is already opened successfully.
*   // Deal is already run successfully.
*
*   // Get misc cashflow.
*   double* misc_flow=get_misc_flow(ptid, FLOW_MISC_FEE_TOTAL);
*   if(NULL==misc_flow)
*   {
*       std::cout<<"The deal does not contain this cashflow field."<<std::endl;
*   }
*
*   close_deal_ex(ptid,&cmo);
*
```

22.3.3.83 int CHASAPI get_monte_carlo_correlation (void ∗ *tid*, MONTE_CARLO_CORRELATION_TYPE *type*, char ∗ *field1*, char ∗ *field2*, double ∗ *correlation*)

This method gets the correlation between two fields. Depending on the correlation type, the fields can be either issuer names or industry names.

Since

2.1.0

Availability CDOnet

Precondition

set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	Indicates which correlation table the user wants to change, refer to enum MONTE_CARLO_CORRELATION_TYPE.

in	*field1*	Name of field1 whose correlation to field2 will be get.
in	*field2*	Name of field2 whose correlation to field1 will be get.
out	*correlation*	The correlation value between field1 and field2.

Return values

0	Success.
-1	Deal not open.
-2	Invalid pointer.
-3	Unrecognized issuer/industry name
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "1776");
*
*      set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*
*      double correlation = 0.0;
*      assert(0 == get_monte_carlo_correlation(pDeal,
      MONTE_CARLO_CORRELATION_INDUSTRY, "DJUSST", "DJUSTA", &correlation));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.84 int CHASAPI get_monte_carlo_global_issuers (void ∗ *tid,* char ∗ *issuer_names[ ],* short *size*)

This method retrieves the list of the issuer names of the portfolio matrix.

Since

2.1.0

Availability CDOnet

Precondition

set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*issuer_names*	A pointer to a client-allocated array of character strings in which the issuer names of the portfolio matrix will be stored. 61 characters should be allocated for each string.
in	*size*	The size of the array that the user has passed down.

Return values

>=0	Success. If issuer_names[] is NULL, it will be the number of the issuer names of the portfolio matrix. If issuer_names is not NULL, it will be the number of issuer names that have been populated.
-1	Deal not opened.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
    *       void* pDeal = NULL;
    *       CMO_STRUCT *pCmo = new CMO_STRUCT();
    *       memset(pCmo, 0, sizeof(*pCmo));
    *       strcpy(pCmo->dealid, "1776");
    *
    *       set_engine_preference(
           PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
    *       assert(0 == open_deal_ex(pDeal, pCmo));
    *
    *       assert(0 == set_simulation_engine(pDeal,
           SIMULATION_MONTE_CARLO));
    *
    *       int numIssuer = get_monte_carlo_global_issuers(pDeal, NULL, 0);
    *       if (numIssur > 0)
    *       {
    *           std::vector<char> issuer_buf(numIssur*61);
    *           std::vector<char*> issuers(numIssur);
    *           for(int i = 0; i<numIssur; ++i)
    *           {
    *               issuers[i] = &issuer_buf[i*61];
    *           }
    *           get_monte_carlo_global_issuers(pDeal, &issuers.front(), numIssuer)
           ;
    *       }
    *
    *       assert(0 == close_deal_ex(pDeal, pCmo));
    *       delete pCmo;
    *       pCmo = NULL;
    *
```

22.3.3.85 int CHASAPI get_monte_carlo_result (void ∗ *tid,* const char ∗ *bondid,* MONTE_CARLO_RESULT ∗ *result*)

This method returns the monte carlo simulation results.

Since

2.1.0

Availability CDOnet, SFW

Precondition

run_monte_carlo_simulation() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	The name of tranche whose result is being requested.
out	*result*	The result being requested.

Return values

0	Success.
-1	Deal not open.
-2	bondid not recognized.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
    *       void* pDeal = NULL;
```

```
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ACE06NC1");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*      assert(0 == run_monte_carlo_simulation(pDeal));
*
*      MONTE_CARLO_RESULT result;
*      assert(0 == get_monte_carlo_result(pDeal, "A1", &result);
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.86 int CHASAPI get_moodys_bond_history (void ∗ *tid*, const char ∗ *bondId*, MOODYS_BOND_HISTORY *bondHistory[ ]*, int *sizeOfHistoryArray*, int *YYYYMM*)

This method gets the historical data of YYYYMM for the specified bondid, if YYYYMM=0 would return all historical data for the specified bondid.

Since

1.6.0

Availability SFW CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondId*	The bond name.
in	*sizeOfHistory-Array*	The size of array pool_history provided.
in	*YYYYMM*	The specified date (of format YYYYMMDD),if YYYYMM=0 would return an array of all historical data for the specified bondid, else it would return a historical data structure of specified YYYYMM.
out	*bondHistory*	The list of bond history. This parameter must be pre-allocated before call this function.

Return values

>=0	Actual size of history array returned.
-1	Error - Deal not opened.
-2	Error - Invalid bond id.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*    void* ptid = NULL;
*    // deal has been opened
*
*    int yyyymm[1] = {0};
*    int avail_num = 0;
*    int ret = get_moodys_bond_history_avail_YYYYMMs(ptid, "A1", yyyymm
       , 1, &avail_num);
*    if(ret < 0)
*    {
```

```
*       // error handling
*   }
*
*   MOODYS_BOND_HISTORY bondhistory[1];
*   ret = get_moodys_bond_history(ptid, "A1", bondhistory, 1, yyyymm[0]); // get
      history of specified YYYYMM for "A1"
*   if(ret < 0)
*   {
*       // error handling
*   }
*
*   MOODYS_BOND_HISTORY* bondhistoryArray = new
      MOODYS_BOND_HISTORY[avail_num];
*   ret = get_moodys_bond_history(ptid, "A1", bondhistoryArray, 1, 0); // get all
      history for "A1"
*   if(ret < 0)
*   {
*       // error handling
*   }
*   delete [] bondhistoryArray;
*   bondhistoryArray = NULL;
*
```

22.3.3.87 int CHASAPI get_moodys_bond_history_avail_YYYYMMs (void ∗ *tid,* const char ∗ *bondId,* int *YYYYMMs[ ],* int *sizeOfYYYYMMs,* int ∗ *numAvailable*)

This method gets the available bond history dates for the specified bondid, the format is in YYYYMM.

Since

1.6.0

Availability SFW CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondId*	The bond name.
in	*sizeOfYYYYM-Ms*	The size of array YYYYMMs provided.
out	*YYYYMMs*	The list of available bond history dates (of format YYYYMMDD). This parameter must be pre-allocated before call this function.
out	*numAvailable*	Total number of available YYYYMMs.

Return values

>=0	Actual number of dates returned.
-1	Error - Deal not opened.
-2	Error - Invalid bond id.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*   void* ptid = NULL;
*   // deal has been opened
*
*   int yyyymm[1] = {0};
*   int avail_num=0;
*   int ret = get_moodys_bond_history_avail_YYYYMMs(ptid, "A1", yyyymm
      , 1, &avail_num);
*   if(ret < 0)
*   {
*       //error handling
*   }
*
```

22.3.3.88 int CHASAPI get_moodys_cmm_scenarios (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* char ∗ *scenario_list[ ]*)

Gets the list of available CMM scenarios of SFW deal.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*scenario_list*	A pointer to the user allocated string array to store the returned CMM scenarios, it need at least 20 characters for each scenario name. If scenario_list is NULL, this function will only return the total count of scenarios.

Return values

>=0	The total count of scenarios.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "CMBS_BOA00002");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       int scenario_count = get_moodys_cmm_scenarios(tid, NULL, NULL);
*       assert(scenario_count > 0);
*
*       char **scenarios = new char*[scenario_count];
*       for (int i = 0; i < scenario_count; ++i)
*        scenarios[i] = new char[20];
*       assert(scenario_count == get_moodys_cmm_scenarios(tid, NULL, scenarios));
*
*       char current_scenario[20];
*       assert(0 == set_current_moodys_cmm_scenario(tid, NULL, scenarios[3]))
       ;
*       assert(0 == get_current_moodys_cmm_scenario(tid, NULL,
       current_scenario));
*       assert(0 == strcmp(scenarios[3], current_scenario));
*
*       assert(0 == run_deal_ex(tid, pCmo));
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

- set_current_moodys_cmm_scenario()
- get_current_moodys_cmm_scenario()

22.3.3.89 int CHASAPI get_moodys_id (const char ∗ *id,* char ∗ *deal,* int *deal_length,* char ∗ *bond,* int *bond_length,* char ∗ *err_buffer,* int *err_length*)

Returns the SDK deal and bond ID for an industry-standard bond identifier. Comparing with get_markit_id1(), this function takes two additional parameters specifying output array sizes so arbitrarily sized deal and bond can be accommodated.

Since

1.4.0

Availability ALL

Parameters

in	*id*	The US or international bond identifier.
out	*deal*	The deal ID. Should pre-allocate at least 20 bytes for this field.
in	*deal_length*	The length of deal.
out	*bond*	The bond ID. Should pre-allocate at least 20 bytes for this field.
in	*bond_length*	The length of bond
out	*err_buffer*	The error message.
in	*err_length*	The length of the error message.

Return values

0	No error, but deal is not found
1	Found deal successfully
-99	Please examine err_buffer for error, deal buffer for deal, bond buffer for bond.

Example:

```
*      const char *cusip;
*      char deal_id[20];
*      char bond_id[20];
*      char error[100]={0};
*
*      cusip = "78443CCA0";
*      get_moodys_id(cusip, deal_id, 20, bond_id, 20, error,100);
*
```

22.3.3.90 long CHASAPI get_moodys_pool_group_id (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int *group_number,* char ∗ *group_id*)

Retrieves Moody's poolgroup ID given a group number.

Since

1.4.0

Availability SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*group_number*	The group number.
out	*group_id*	A pointer to the user allocated string for group id.

Return values

0	Success
-99	Error - For details call get_deal_error_msg()

Note

For CHS deals, method throws exception "not implemented in CHS engine"

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "NSC0301");
*
*       open_deal_ex(tid,pCmo);
*
*       char grp_id[12];
*       get_moodys_pool_group_id(tid,NULL,1,grp_id);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.91 int CHASAPI get_moodys_pool_history (void ∗ *tid*, int *groupNumber*, MOODYS_POOL_HISTORY *poolHistory[ ]*, int *sizeOfHistoryArray*, int *YYYYMM*)

This method gets the historical data of YYYYMM for the specified pool group, if YYYYMM=0 would return all historical data for the specified pool group.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

`in`	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
`in`	*groupNumber*	The 1-based group number of the reqeusted pool group.
`in`	*sizeOfHistory-Array*	The size of array pool_history provided.
`in`	*YYYYMM*	The specified date (of format YYYYMM),if YYYYMM=0 would return all historical data for the specified pool group.
`out`	*poolHistory*	The list of pool history. This parameter must be pre-allocated before call this function.

Return values

>=0	Actual size of history array returned.
-1	Error - Deal not opened.
-2	Error - Invalid pool group number.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*       void* ptid = NULL;
```

```
 *      // deal has been opened
 *
 *      int yyyymm[1] = {0};
 *      int avail_num=0;
 *      int ret = get_moodys_pool_history_avail_YYYYMMs(ptid, 1, yyyymm
       , 1, &avail_num);
 *      if(ret < 0)
 *      {
 *          // error handling
 *      }
 *
 *      MOODYS_POOL_HISTORY poolhistory[1];
 *      int actual_size=0;
 *      ret = get_moodys_pool_history(ptid, 1, poolhistory, 1, yyyymm[0],&actual_size
       );//get history of specified YYYYMM
 *      if(ret < 0)
 *      {
 *          // error handling
 *      }
 *
 *      MOODYS_POOL_HISTORY* poolhistoryArray = new
       MOODYS_POOL_HISTORY[avail_num];
 *      int actual_size=0;
 *      ret = get_moodys_pool_history(ptid, 1, poolhistoryArray, 1, 0,&actual_size);
       //get all history for the group 1
 *      if(ret < 0)
 *      {
 *          // error handling
 *      }
 *      delete [] poolhistoryArray;
 *      poolhistoryArray = NULL;
 *
```

22.3.3.92 int CHASAPI get_moodys_pool_history_avail_YYYYMMs (void ∗ *tid,* int *groupNumber,* int *YYYYMMs[ ],* int *sizeOfYYYYMMs,* int ∗ *numAvailable*)

This method gets the available pool history dates for the specified pool group, the format is in YYYYMM.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*groupNumber*	The 1-based group number of the reqeusted pool group.
in	*sizeOfYYYYM-Ms*	The size of array YYYYMMs provided.
out	*YYYYMMs*	The list of available pool history dates (of format YYYYMMDD). This parameter must be pre-allocated before call this function.
out	*numAvailable*	Total number of available YYYYMMs.

Return values

>=0	Actual number of dates returned.
-1	Error - Deal not opened.
-2	Error - Invalid pool group number.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
 *      void* ptid = NULL;
```

```
*       // deal has been opened
*
*       int yyyymm[1] = {0};
*       int avail_num=0;
*       int ret = get_moodys_pool_history_avail_YYYYMMs(ptid, 1, yyyymm
      , 1, &avail_num);
*       if(ret < 0)
*       {
*           //error handling
*       }
*
```

22.3.3.93 long CHASAPI get_moodys_ssfa_calc (void ∗ *tid,* const char ∗ *bondid,* MOODYS_SSFA_CALC ∗ *ssfa_calc*)

Gets Moodys SSFA calculation for a bond.

New feature Subject to change

Since

2.0.0

Availability SFW, CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	A pointer to the name of the bond.
out	*ssfa_calc*	A pointer to the user allocated structure holding bond ssfa calculation.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - For details call get_deal_error_msg()

Note

Currently supported asset classes (based on MARKIT_DEAL_INFO::asset_type):

Asset Type	MARKIT_DEAL_INFO::asset_type string matches
RMBS	MBS∗, HELOC, Home_Equity, UK_RMBS, Whole_Loan
CMBS	CMBS∗
SLABS	Student_Loan∗
ABS	ABS∗, Auto∗, Equip, FloorPlan, Manufactured_Housing, PPLN, SBA, StructNote
CDO	CDO deals

Example:

```
*       void* ptid = NULL;
*       CMO_STRUCT deal={};
*       strcpy(deal.dealid,"CMBS_BOA06001");
*
*       //open deal
*       open_deal_ex(ptid, &deal);
*
```

```
 *      // SSFA variable
 *      MOODYS_SSFA_CALC ssfa_calc={};
 *      // get ssfa calc for bond "X"
 *      get_moodys_ssfa_calc(ptid, "X", &ssfa_calc);
 *
 *      // close deal
 *      close_deal_ex(ptid, &deal);
 *
```

22.3.3.94 int CHASAPI get_mpa_economy_date (void ∗ *tid,* int ∗ *year,* int ∗ *quarter*)

This function will return the MPA model economy date(year and quarter).

Since

2.0.0

Availability All

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*year*	The year of the MPA economy date, e.g. 2013.
out	*quarter*	The quarter of the MPA economy date, the value of quarter should be an integer in the range of [1, 4].

Return values

0	Success.
-2	Error - Invalid year or quarter pointers.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
 *      void* pDeal = NULL;
 *      CMO_STRUCT *pCmo = new CMO_STRUCT();
 *      memset(pCmo, 0, sizeof(*pCmo));
 *      strcpy(pCmo->dealid, "ACE06NC1");
 *      set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
 *
 *      int year = 0, quarter = 0;
 *      assert(0 == get_mpa_economy_date(pDeal, &year, &quarter)); // e.g. year = 2013,
        quarter = 4
 *
 *      assert(0 == open_deal_ex(pDeal, pCmo));
 *      assert(0 == set_moodys_credit_model_settings(pDeal,
       MOODYS_MPA_SETTINGS, false));
 *      assert(0 == run_deal_ex(pDeal, pCmo));
 *
 *      assert(0 == close_deal_ex(pDeal, pCmo));
 *      delete pCmo;
 *      pCmo = NULL;
 *
```

22.3.3.95 int CHASAPI get_mpa_scenarios (void ∗ *tid,* char ∗ *scenario_list[]*)

This function will get array of scenarios names available in current MPA module.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*scenario_list*	A array of char pointers that points to current names of MPA scenarios, it need at least 100 characters for each scenario name. Pass NULL for scenario_list to get just the number of scenarios.

Return values

>0	Number of scenarios found.
0	None scenarios found.
-1	Error - Deal not open.
-2	Error - Invalid MPA analysis type.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ACE06NC1");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
     MOODYS_MPA_SETTINGS, false));
*      assert(0 == set_mpa_analysis_type(pDeal,
     MPA_MEDC_SINGLE_PATH));
*
*      int scenario_count = get_mpa_scenarios(pDeal, NULL);
*      assert(scenario_count > 0);
*
*      std::vector<char *> scenario_list(scenario_count);
*      std::vector<char> scenario_strs(scenario_count*100);
*
*      for (int i = 0; i < scenario_count; ++i)
*          scenario_list[i] = &scenario_strs[i*100];
*
*      assert(scenario_count == get_mpa_scenarios(pDeal, &scenario_list[0]));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

- set_current_mpa_scenario()
- get_current_mpa_scenario()

22.3.3.96 MOODYS_POOL_INFO∗ CHASAPI get_next_collat_ex (void ∗ *tid,* void ∗ *collat_iterator*)

This function gets next collateral information using iterator obtained from calling obtain_collat_iterator_ex(). When iterator goes to the end of the collateral set, it returns NULL.

Since

3.0.0

Availability CDOnet, SFW, CHS

Precondition

open_deal_ex() has been called.
obtain_collat_iterator_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier
in	*collat_iterator*	Pointer to collateral iterator returned by calling obtain_collat_iterator_ex().

Return values

Pointer	Pointer to the next collateral information
0	No more information left

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "STATICLO");
*
*       set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       MOODYS_POOL_INFO* coll_info =0;
*       void* coll_it =obtain_collat_iterator_ex(tid, 0);
*       if(coll_it == 0)
*       {
*           std::cout << "Failure to start collat iteration " <<
      get_deal_error_msg(tid) << std::endl;
*       }
*       while(coll_info =  get_next_collat_ex(tid,coll_it))
*       {
*           // do what you need with collateral
*       }
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

Function returns pointers to collateral information allocated by the API. These pointers will be valid until deal is closed or another call to obtain_collat_iterator_ex() function is made using the same parameters. The iterator will be released when close_deal_ex() is called. The iterator will be overwritten when obtain_collat_iterator_-ex() is called again.

22.3.3.97 int CHASAPI get_optional_redemption_date (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* char ∗ *date*)

This method returns the optional redemption date of a deal.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*date*	A pointer to a null-terminated string (of format MM/DD/YY). This parameter must be pre-allocated with at least 11 characters.

Return values

0	The issued date of the requested pool group has been obtained successfully.
-1	Error - Deal not opened.
-2	Error - Invalid pointer to date.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*     void* tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT;
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "PARAGONM11");
*
*     set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*     assert(0 == open_deal_ex(tid, pCmo));
*
*     char optionalRedemptionDate[11] =  {0};
*     assert(0 == get_optional_redemption_date(tid, NULL,
      optionalRedemptionDate));
*
*     assert(0 == run_deal_ex(tid, pCmo));
*     assert(0 == close_deal_ex(tid, pCmo));
*     delete pCmo;
*     pCmo = NULL;
*
```

22.3.3.98 int CHASAPI get_pa_default_pool_data (void ∗ *tid*, const char ∗ *paraName*, char ∗ *value*, int & *len*)

get the default pool data and other settings set by calling setPAModelDefaultSettings.

Since

2.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().
set_pa_default_pool_data has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*paraName*	The name of the parameter: • "WALoanAge" • "WARemainingTerm" • "WAFICO" • "WACoupon" • "WAPrepayPenaltyTerm" • "AverageOriginalLoanAmount" • "WAOriginalLTV" • "PurposePurchase" • "PurposeRefi" • "OccupancyOwner" • "OccupancySecondHome" • "OccupancyInvestor" • "Property1Unit" • "Property24Unit" • "OriginatorThirdParty" • "OriginatorRetail" • "HARP1" • "HARP2" • "FHA" • "WACAtIssuance" • "WAFixedRatePeriod" • "ArmIndex" • "WAResetInterval" • "WALifetimeCap" • "WALifetimeFloor" • "WAPeriodicCap" • "WAInitialCap" • "WAMargin" • "HistoricalPrepaymentPeriod" • "HistoricalPrepaymentRate" • "Factor" • "Rate30" • "Rate60" • "Rate90"
		• "CPR" • "CDR"

in	*len*	The buffer length of value. If the buffer length is not enough, needed length will be returned through this parameter.

Return values

0	Success.
-1	Deal not open.
-2	paraName is invalid.
-3	PA model is not setup.
-4	Setting is not found.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*       assert(0 == set_pa_default_pool_data(pDeal,"WAFICO","600.0");
*
*       int buflen = 50;
*       char value[buflen]={0};
*       assert(0 == get_pa_default_pool_data(pDeal,"WAFICO",value,buflen);
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.99 int CHASAPI get_pa_economy_date (void ∗ *tid,* int ∗ *year,* int ∗ *quarter*)

This function will return the PA model economy date(year and quarter).

Since

2.0.0

Availability All

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*year*	The year of the PA economy date, e.g. 2013.
out	*quarter*	The quarter of the PA economy date, the value of quarter should be an integer in the range of [1, 4].

Return values

0	Success.
-2	Error - Invalid year or quarter pointers.
-3	Error - Current credit model is not PA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
```

```
 *      set_engine_preference(
        PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
 *
 *      int year = 0, quarter = 0;
 *      assert(0 == get_pa_economy_date(pDeal, &year, &quarter)); // e.g. year = 2013,
        quarter = 4
 *
 *      assert(0 == open_deal_ex(pDeal, pCmo));
 *      assert(0 == set_moodys_credit_model_settings(pDeal,
        MOODYS_PA_SETTINGS, false));
 *      assert(0 == run_deal_ex(pDeal, pCmo));
 *
 *      assert(0 == close_deal_ex(pDeal, pCmo));
 *      delete pCmo;
 *      pCmo = NULL;
 *
```

22.3.3.100 int CHASAPI get_pa_model_type (void ∗ *tid,* char ∗ *pa_model_type,* int *pa_avail_vector[ ],* int ∗ *avail_vector_num*)

The function will retrieve pa support type for the deal.

New feature Subject to change

Since

2.7.0

Availability CHS,SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*pa_model_type*	PA model type for the deal, should pre-allocate at least 50 bytes for this field.
out	*pa_avail_vector*	PA output available vector type for the deal, should pre-allocate int array of length NUM_PA_POOL_VECTOR_TYPES.
out	*avail_vector_-num*	Number of available PA vector type.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
 *     void *pDeal = NULL;
 *     //deal has been opened
 *
 *     char pa_model_type[50] = {0};
 *     int pa_avail_vector[NUM_PA_POOL_VECTOR_TYPES];
 *     int avail_num = 0;
 *     int ret = get_pa_model_type(pDeal, pa_model_type, pa_avail_vector, &avail_num);
 *     if (0 != ret)
 *     {
 *         // error handle
 *     }
 *
```

22.3.3.101 int CHASAPI get_pa_scenarios (void ∗ *tid,* char ∗ *scenario_list[]*)

This function will get PA model available scenarios.

Since

2.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*scenario_list*	The list of scenarios. If this parameter is NULL, this function will return with number of available scenarios. The memory of scenario_list should be allocated by user. The lengh of each element of scenario_list should be 100.

Return values

>=0	Success. Number of available scenarios.
-1	Deal not opened.
-2	Error - Invalid pointer.
-3	Error - PA model is not setup.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*
*       int scenario_count = get_pa_scenarios(pDeal, NULL);
*       assert(scenario_count > 0);
*
*       std::vector<char *> scenario_list(scenario_count);
*       std::vector<char> scenario_strs(scenario_count*100);
*       for (int i = 0; i < scenario_count; ++i)
*           scenario_list[i] = &scenario_strs[i*100];
*
*       assert(scenario_count == get_pa_scenarios(pDeal, &scenario_list[0]));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.102 double∗ CHASAPI get_pa_vector (void ∗ *tid,* int *group_number,* PA_POOL_VECTOR_TYPE *identifier*)

This method gets pa model generated vectors

New feature Subject to change

Since

2.0.1

Availability CHS,SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().
run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*group_number*	The group number of collateral.
in	*identifier*	The indentifier of vector type.

Return values

The	pointer to the vector

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       double* v = get_pa_vector(pDeal,0,PA_VECTOR_CHANGEOFF);
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

Only PA_CPR is supported in CHS.

22.3.3.103 int CHASAPI get_prospectus_prepayment_curves (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, PPC_STRUCT *all_PPCs[]*, int *size*, int ∗ *num_curves*)

Gets prospectus prepayment curves from deal.

Since

3.7

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*all_PPCs*	A client-allocated array of PPC_STRUCT for prospectus prepayment curves information.
in	*size*	The length of the array all_PPCs that the user has passed.
out	*num_curves*	Total number of available curves.

Return values

>=0	Success. Actual number of prospectus prepayment curves returned.
-1	Error - Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Note

If pass NULL, the function will return the number of prospectus prepayment curves.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "NL2017-B");
*
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       int num_curves = 0;
*       assert(0 == get_prospectus_prepayment_curves(pDeal, NULL, NULL, 0, &
      num_curves));
*
*       PPC_STRUCT *all_ppc = new PPC_STRUCT[num_curves];
*       assert(0 <= get_prospectus_prepayment_curves(pDeal, NULL, all_ppc,
      num_curves, &num_curves));
*
*       delete[] all_ppc;
*       all_ppc = NULL;
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.104 int CHASAPI get_reinv_recovery_rate (void ∗ *tid,* long *loan_num,* double ∗ *recovery_rate*)

This method retrieves reinvestment recovery rate of a specified reinvestment loan.

Since

2.0.1

Availability CDOnet

Precondition

open_deal_ex() has been called.
The current credit model has been set to SEDF with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_num*	Ordinal collateral number in a deal.
out	*recovery_rate*	A pointer to a client-allocated of double to store recovery rate data.

Return values

0	Success.
-1	Deal not open.
-2	Invalid loan_num or tid.
-3	Current credit model is not Stress EDF.
-4	Invalid pointer.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "1776");
*
*      set_engine_preference(
     PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      double rec_rate = 0.0;
*      assert(0 == get_reinv_recovery_rate(pDeal, 1, &rec_rate));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.105 int CHASAPI get_reinv_weighted_avg_pd (void ∗ *tid*, long *loan_num*, double *pd[ ]*)

This method retrieves weighted average default probabilities of a specified reinvestment loan.

Since

2.0.1

Availability CDOnet

Precondition

open_deal_ex() has been called.
The current credit model has been set to SEDF with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_num*	Ordinal collateral number in a deal.
out	*pd*	A pointer to a client-allocated array of double to store EDF data. At least 5 elements should be allocated.

Return values

0	Success.
-1	Deal not open.
-2	Invalid loan_num or tid.
-3	Current credit model is not Stress EDF.
-4	Invalid EDF data array.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "1776");
*
*       set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       double pd[5]={0.0};
*       assert(0 == get_reinv_weighted_avg_pd(pDeal, 1, pd));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.106 int CHASAPI get_required_index_codes (void ∗ *tid,* const char ∗ *currency,* int ∗ *rate_codes,* int *size_of_array_codes*)

Returns number of market rate indices under the specified currency used by currently open deal (includes bonds, collateral, hedges, accounts, and waterfall script) and populates rate_codes array with the list of index rate codes that are used.

New feature Subject to change

Since

2.0.0

Availability ALL

Precondition

open_deal_ex() was called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The specified currency name.
out	*rate_codes*	The list of index rates codes used in the deal.
in	*size_of_array_-codes*	The size of the user allocated array rate_codes.

Return values

>=0	No error

-1	Error - Deal not opened
-2	Error - Invalid ISO currency name
-3	Error - Invalid rate_codes pointer
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void* ptid=NULL;
*     // Deal is open
*
*     int required_rates[MAX_INDEX_TYPES_EX]={0};
*     //the codes of index rates will return in 'required_rates'.
*     int actual_count = get_required_index_codes(ptid, "EUR", required_rates,
      MAX_INDEX_TYPES_EX);
*     if (actual_count < 0)
*     {
*         // error handle
*     }
*
```

Note

For CHS engine, the currency just support the "USD".

22.3.3.107 int CHASAPI get_sub_trigger_info (void * *tid*, const char * *reremic_deal_id_or_null*, const char * *sub_trigger_name*, char * *sub_trigger_type*, char * *sub_trigger_operator*, double * *current_level*, double * *threshold*, SBYTE * *status*, BOOLYAN * *curable*, SBYTE * *override_type*, int * *override_date*)

This method return the details of the current and projected status of a sub-trigger.

Since

2.0.1

Availability SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*sub_trigger_-name*	The case-sensitive name of the sub-trigger whose trigger information are re-quested.
out	*sub_trigger_type*	Trigger type of the requested sub-trigger. 1025 characters should be allocated for the string.
out	*sub_trigger_-operator*	Trigger operator of the requested sub-trigger. 10 characters should be allo-cated for the string.
out	*current_level*	A pointer to a client-allocated array in which the current and projected values of the level of the requested sub-trigger will be stored. The size of the array should be of MAX_PERIODS.

out	*threshold*	A pointer to a client-allocated array in which the current and projected values of the threshold of the requested sub-trigger will be stored. The size of the array should be of MAX_PERIODS.
out	*status*	A pointer to a client-allocated array in which the current and projected values of the status of the sub-trigger will be stored. The size of the array should be of MAX_PERIODS.
out	*curable*	This flag indicates if the sub-trigger is curable or not.
out	*override_type*	The flag indicates if the sub-trigger is overridden or will be overridden. 0 for no override, 1 for always "no" and 2 for always "yes".
out	*override_date*	Override date of the sub-trigger if available. The format is "YYYYMMDD".

Return values

0	Success.
-1	Deal not open.
-2	Invalid pointer
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       //Deal has been run
*
*       char trigger_type[1025] = {0};
*       char trigger_operator[10] = {0};
*       double threshold[MAX_PERIODS] = {0.};
*       signed char status[MAX_PERIODS] = {0};
*       double currentlevel[MAX_PERIODS] = {0.};
*       BOOLYAN curable = 0;
*       signed char override_type = 0;
*       int override_date = 0;
*       int ret = get_sub_trigger_info(pDeal, NULL, "PRO-RATA-1", trigger_type,
      trigger_operator, currentlevel, threshold, status, &curable, &override_type, &override_date);
*       if(ret != 0)
*       {
*           //Error handle
*       }
*
```

22.3.3.108 int CHASAPI get_trigger_avail_ex (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, char ∗ *trigger_names[ ]*, char ∗ *trigger_descs[ ]*, int ∗ *num_sub_triggers*, int *size*)

This method returns a list of high level trigger information(trigger names, trigger descriptions, number of sub-triggers that a trigger has) of an SFW deal.

Since

2.0.1

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
out	*trigger_names*	A pointer to a client-allocated array of character strings in which the names of the triggers will be stored. 21 characters should be allocated for each string.
out	*trigger_descs*	A pointer to a client-allocated array of character strings in which the descriptions of the triggers will be stored. 1025 characters should be allocated for each string.
out	*num_sub_-triggers*	A pointer to a client-allocated array of integer in which the sum of the sub-triggers for each main triggers.
in	*size*	The length of the arrays that the user has passed down.

Return values

>0	Number of triggers whose information has been returned.
=0	Success, but deal haven't triggers
-1	Deal not open.
-2	Invalid pointer
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       //deal has been opened
*
*       char trg_name[5][21] = {0};
*       char trg_dec[5][1025] = {0};
*       int num_subtriggers[5] = {0};
*       std::vector<char*> vec_names(5);
*       std::vector<char*> vec_decs(5);
*       for(int i = 0; i<5; i++)
*       {
*           vec_names[i] = trg_name[i];
*           vec_decs[i] = trg_dec[i];
*       }
*       int ret = get_trigger_avail_ex(pDeal, NULL, &vec_names.front(), &vec_decs.front(
       ), num_subtriggers, 5);
*       if(ret < 0)
*       {
*           //Error handle
*       }
*
```

Note

Pass NULL for trigger_names ,trigger_descs and num_sub_triggers to get just the number of triggers.

22.3.3.109 long CHASAPI get_trustee_loan_id (void * *tid*, const char * *reremic_deal_id_or_null*, int *loan_number*, char * *trustee_loan_id*)

Get trustee loan ID.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*loan_number*	The 1-based index of the loan.
out	*trustee_loan_id*	A pointer to the user allocated string for account id, this must be pre-allocated with at least 20 characters.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - For details call get_deal_error_msg()

Example:

```
*       void* ptid = NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"CMBS_BOA06001");
*
*       //open deal
*       open_deal_ex(ptid, &deal);
*
*       char trustee_loan_id[20];
*       char *reremic_deal_id_or_null = NULL;
*
*       assert(0 == get_trustee_loan_id(ptid, reremic_deal_id_or_null, 1, trustee_loan_id
       ));
*       assert("C10071038" == trustee_loan_id);
*
*       assert(0 == get_trustee_loan_id(ptid, reremic_deal_id_or_null, 2, trustee_loan_id
       ));
*       assert("C10071071" == trustee_loan_id);
*
*       close_deal_ex(ptid, &deal);
*
```

22.3.3.110 long CHASAPI ignore_asset_nonpayment_term (void * *tid,* bool *val*)

This function enable user to specify if their input status vectors overwrite the asset level status.

If set a TRUE flag, API will use the status vector set by user and ignoring the asset level non-payment status. If set a FALSE flag, API will use both status vector and asset level non-payment status.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*val*	• If set with TRUE, API will apply the status vector set by user and ignore the status from asset level. • If set with FALSE, API will both use vectors set by user and non-payment status on asset level.

Return values

0	Success
-1	Error - Deal not opened
-99	Error - For details call get_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SMS2000-A");
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       // case 1: no ignore_asset_nonpayment_term(), not set status vectors
*       assert(0 == run_deal_ex(tid, pCmo));
*
*       // case 2: ignore_asset_nonpayment_term(false), not set status vectors, equals to case 1
*       assert(0 == ignore_asset_nonpayment_term(tid, false));
*       assert(0 == run_deal_ex(tid, pCmo));
*
*       // case 3: ignore_asset_nonpayment_term(false), set status vectors
*
*       // set forbearance vector
*       #define MAX_MONTHS 500
*       double adRates[MAX_MONTHS];
*
*       std::fill_n(adRates, MAX_MONTHS, 0.0);
*       std::fill_n(adRates+1, 30, 15.0/100);    // adRates[0] is not been used
*
*       short is_vector = MAX_MONTHS;
*       double *pval = adRates;
*       long loan_num = -1;
*       BOOLYAN set_sup_remic = false;
*       assert(0 == set_forbearance_rates(tid, is_vector, pval, loan_num, set_sup_remic
      ));
*       assert(0 == ignore_asset_nonpayment_term(tid, false));
*       assert(0 == run_deal_ex(tid, pCmo));
*
*       // case 4: ignore_asset_nonpayment_term(true), set status vectors
*       std::fill_n(adRates, MAX_MONTHS, 0.0);
*       std::fill_n(adRates+1, 30, 15.0/100);    // adRates[0] is not been used
*       assert(0 == set_forbearance_rates(tid, is_vector, pval, loan_num, set_sup_remic
      ));
*       assert(0 == ignore_asset_nonpayment_term(tid, true));
*       assert(0 == run_deal_ex(tid, pCmo));
*
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

- set_forbearance_rates()
- set_deferment_rates()
- set_grace_rates()

22.3.3.111 int CHASAPI install_collat_assump_cb_ex1 (void * *tid,* COLLAT_ASSUMP_CB_EX1 *collat_assump_cb_ex1*)

refer to install_collat_assump_cb() and install_collat_assump_cb_ex(), install_collat_assump_cb_ex1() add a param MOODYS_POOL_INFO.

Since

2.9.3

Availability CDOnet, SFW, CHS

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*collat_assump_-cb_ex1*	The user provided call back function.

Return values

0	Success.
-99	Error - use get_deal_error_msg() function to obtain text of error.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       open_deal_ex(tid, pCmo);
*
*       COLLAT_ASSUMP_CB_EX1 collat_assump_cb_ex1;
*       install_collat_assump_cb_ex1(tid, collat_assump_cb_ex1);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.112 double∗ CHASAPI load_MA_rate (const char ∗ *file_path,* int *yyyymmdd,* const char ∗ *ma_scenario,* const char ∗ *currency,* short ∗ *idx*)

Get MA interest rates from MWSA table before open deal. The function looks back as many as 7 calendar days to avoid data availability gaps from weekends and holidays.

Since

3.7.0

Availability CDOnet, SFW, CHS

Parameters

in	*file_path*	Directory where the MWSA DB resides.
in	*yyyymmdd*	User input date with format YYYYMMDD. In case the MWSA DB of YYYYMM-DD is not available, look back as many as 7 days.
in	*ma_scenario*	User input scenario type.
in	*currency*	The ISO currency code of the specified interest rate (e.g., "USD").
in	*idx*	The rate index type. ∗idx must be one of: enums of INDEX_TYPE enums of INDEX_TYPE_EX.

Return values

NULL	- The requested index rate was not retrieved successfully.
Other	- A pointer to an array which stores the interest rates in decimal from MWSA DB.

Example:

```
*       const char* file_path = "path";
*       int yyyymmdd = 20170602;
*    const char* currency_sek = "SEK";
*   short idx_libor3m = LIBOR_3;
*   double* sek_libor3m_rates = load_MA_rate(file_path, yyyymmdd, "BL", currency_sek, &
      idx_libor3m);
*
```

22.3.3.113 int CHASAPI load_MA_rates (void ∗ *tid,* int *yyyymmdd,* const char ∗ *ma_scenario*)

This method gets the available MA rate shifts scenarios list for the specified date yyyymmdd. The function looks back as many as 7 calendar days to avoid data availability gaps from weekends and holidays.

Since

3.7.0

Availability SFW, CHS, CDONET

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*yyyymmdd*	The specified date (of format YYYYMMDD).
in	*ma_scenario*	the scenario type indicating which scenario to be set for MA rate shifts corresponding to get_MA_rate_shifts_scenarios().

Return values

>0	Actual date to be loaded.
-1	Error - Deal not opened or Invalid date input.
-2	Error - MA/MEDC Rates for yyyymmdd not found.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*      void* ptid = NULL;
*      // deal has been opened
*
*      int ret = load_MA_rates(ptid, 20161225, "BL");
*      if(ret < 0)
*      {
*          // error handling
*      }
*
```

22.3.3.114 double∗ CHASAPI load_MWSA_rate (const char ∗ *file_path,* int *yyyymmdd,* const char ∗ *currency,* short ∗ *idx*)

Get interest rates from MWSA DB before open deal. The function looks back as many as 7 calendar days to avoid data availability gaps from weekends and holidays.

Since

3.3.0

Availability CDOnet, SFW, CHS

Parameters

in	*file_path*	Directory where the MWSA DB resides.
in	*yyyymmdd*	User input date with format YYYYMMDD. In case the MWSA DB of YYYYMM-DD is not available, look back as many as 7 days.
in	*currency*	The ISO currency code of the specified interest rate (e.g., "USD").
in	*idx*	The rate index type. ∗idx must be one of: enums of INDEX_TYPE enums of INDEX_TYPE_EX.

Return values

NULL	- The requested index rate was not retrieved successfully.
Other	- A pointer to an array which stores the interest rates in decimal from MWSA DB.

Example:

```
*       const char* file_path = "path";
*       int yyyymmdd = 20170602;
*    const char* currency_sek = "SEK";
*    short idx_libor3m = LIBOR_3;
*    double* sek_libor3m_rates = load_MWSA_rate(file_path, yyyymmdd, currency_sek, &
       idx_libor3m);
*
```

22.3.3.115 int CHASAPI load_MWSA_rates (void ∗ *tid,* int *yyyymmdd,* BOOLYAN *load_forward_curves*)

This method gets the MWSA rates for the specified date yyyymmdd and sets rates to deal. The function looks back as many as 7 calendar days to avoid data availability gaps from weekends and holidays.

Since

3.0.0

Availability SFW, CHS, CDONET

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*yyyymmdd*	The specified date (of format YYYYMMDD).
in	*load_forward_-curves*	If true, load the forward curves of applicable rate types.

Return values

>0	Actual date to be loaded.
-1	Error - Deal not opened or Invalid date input.
-2	Error - MWSA Rates for yyyymmdd not found.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*       void* ptid = NULL;
*       // deal has been opened
*
*       int ret = load_MWSA_rates(ptid, 20161225, true);
*       if(ret < 0)
*       {
*           // error handling
*       }
*
```

22.3.3.116 void∗ CHASAPI obtain_collat_iterator_ex (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null*)

This function obtains a pointer to the internal to the WSA API collateral iterator. This pointer should be passed to consecutive calls to get_next_collat_ex() to retrieve collateral information.

Keep in mind that the second call to this function for the same deal will invalidate all the collateral pointers retrieved from the WSA API during the first call.

Since

3.0.0

Availability CDOnet, SFW, CHS

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier.
in	*reremic_deal_id-_or_null*	Pass 0 for main deal or remic name for collateral of the child deal

Return values

Pointer	to be passed to get_next_collat_ex() function
0	Error

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "STATICLO");
*
*      set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(tid, pCmo));
*
*      MOODYS_POOL_INFO* coll_info =0;
*      void* coll_it =obtain_collat_iterator_ex(tid, 0);
*      if(coll_it == 0)
*      {
*          std::cout << "Failure to start collat iteration " <<
      get_deal_error_msg(tid) << std::endl;
*      }
*      while(coll_info =  get_next_collat_ex(tid,coll_it))
*      {
*          // do what you need with collateral
*      }
*
*      assert(0 == close_deal_ex(tid, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.117 int CHASAPI price_loan (void ∗ *tid,* int *loan_number,* PRICING_ANCHORS *anchorType,* double *anchorValue,* PRICING_RESULTS ∗ *results*)

This function calculates cashflow analytics for a given loan. It should be called after running cashflow

Since

3.2

Availability CDOnet, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*anchorType*	Type of anchor for pricing, available options: PRICE, YIELD, or DM
in	*anchorValue*	Value of the provided pricing anchor
out	*results*	Calculated loan analytics

Return values

0	Success.
-1	Error - Deal not open
-2	Error - Current calculation level is not CALC_LEVEL_FULL_WITH_LOAN
-3	Error - run_deal_ex() has not been called
-99	Error - Other error, use get_deal_error_msg() to see details.

Example:

```
*     void* tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT();
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "DRYDEN34");
*
*     set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*     assert(0 == open_deal_ex(tid, pCmo));
*
*     set_deal_calc_level(pDeal, CALC_LEVEL_FULL_WITH_LOAN, 1);
*     assert(0 == run_deal_ex(tid,pCmo));
*
*     double dAnchor             = 2.0000;
*     PRICING_ANCHORS anchorType = YIELD;
*     PRICING_RESULTS results;
*     memset(&results, 0x00, sizeof(PRICING_RESULTS));
*     int nRet = price_loan(tid, 1, anchorType, dAnchor, &results);
*     if(nRet !=0)
*     {
*         //error handle;
*     }
*
*     assert(0 == close_deal_ex(tid, pCmo));
*     delete pCmo;
*     pCmo = NULL;
*
```

22.3.3.118 int CHASAPI price_loan_ex (void ∗ *tid*, int *loan_number*, LOAN_PRICING_INPUT *pricing_param_input*, PRICING_RESULTS ∗ *results*)

This extended function calculates cashflow analytics for a given loan. It should be called after running cashflow

Since

3.2

Availability CDOnet, SFW

Precondition

run_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*pricing_param_-input*	structure of price loan input information.
out	*results*	Calculated bond analytics

Return values

0	Success.
-1	Error - Deal not open
-2	Error - Current calculation level is not CALC_LEVEL_FULL_WITH_LOAN
-3	Error - invalid loan number
-4	Error - invalid rate index
-99	Error - Other error, use get_deal_error_msg() to see details.

Example:

```
*     void* tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT();
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "DRYDEN34");
*
*     set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*     assert(0 == open_deal_ex(tid, pCmo));
*
*     set_deal_calc_level(pDeal, CALC_LEVEL_FULL_WITH_LOAN, 1);
*     assert(0 == run_deal_ex(tid,pCmo));
*
*     LOAN_PRICING_INPUT pricing_input;
*     // set informations for the pricing_input
*     PRICING_RESULTS results;
*     memset(&results, 0x00, sizeof(PRICING_RESULTS));
*     int nRet = price_loan_ex(tid, 1, pricing_input, &results);
*     if(nRet !=0)
*     {
*         //error handle;
*     }
*
*     assert(0 == close_deal_ex(tid, pCmo));
*     delete pCmo;
*     pCmo = NULL;
*
```

22.3.3.119 int CHASAPI remove_simulation_cashflow_populated_limit (void ∗ *tid*, BOOLYAN *flag*)

This method is to set the flag of simulation cashflow populated limit,If flag is set to TRUE, all simulation paths of cash flows can be retrieved by function get_collateral_flow_sim and get_bond_flow_sim. If flag is set to FALSE, the paths of cash flows populated can not be greater than 100. By default, flag is set to FALSE.

Since

3.6

Availability SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*flag*	The flag of simulation cashflow populated limit.

Return values

0	No error.
-1	Error - Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == remove_simulation_cashflow_populated_limit(
       pDeal, true));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.120 int CHASAPI replace_pa_pool_data (void ∗ *tid*, int *poolID*, const char ∗ *paraName*, const char ∗ *value*)

Replace pool data of PA model.

New feature Subject to change

Since

2.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*poolID*	The 1-based index of the pool or -1 to apply to all pool in the deal.
in	*paraName*	The name of the parameter user want to replace: • "WALoanAge" • "WARemainingTerm" • "WAFICO" • "WACoupon" • "WAPrepayPenaltyTerm" • "AverageOriginalLoanAmount" • "WAOriginalLTV" • "PurposePurchase" • "PurposeRefi" • "OccupancyOwner" • "OccupancySecondHome" • "OccupancyInvestor" • "Property1Unit" • "Property24Unit" • "OriginatorThirdParty" • "OriginatorRetail" • "HARP1" • "HARP2" • "FHA" • "WACAtIssuance" • "WAFixedRatePeriod" • "ArmIndex" • "WAResetInterval" • "WALifetimeCap" • "WALifetimeFloor" • "WAPeriodicCap" • "WAInitialCap" • "WAMargin" • "HistoricalPrepaymentPeriod" • "HistoricalPrepaymentRate" • "Factor" • "Rate30" • "Rate60" • "Rate90" • "CPR" • "CDR"

Return values

0	Success.
-1	Deal not opened.
-2	Parameter is NULL.
-3	Current mode is not PA mode
-4	Parameter is not supported.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       open_deal_ex(tid, pCmo);
*       set_moodys_credit_model_settings(tid,
      MOODYS_PA_SETTINGS, false);
*
*       replace_pa_pool_data(tid,1,"PurposePurchase","0");
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.121 int CHASAPI run_default_probability_distribution (void ∗ *tid*)

This method runs the default probability distribution simulation using preset setting.

After calling this method, all the cashflows for the scenarios will be generated and stored in the memory.

User can use methods get_dpd_results(), get_bond_flow_sim() and get_collateral_flow_sim() to get the corresponding simulation results.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid tid.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF00001");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*      DPD_ASSUMPTION dpdAssumption;
*      memset(&dpdAssumption, 0, sizeof(dpdAssumption));
*      dpdAssumption.distribution = DPD_DISTRIBUTION_LOGNORMAL;
*      dpdAssumption.scenario_type = 1;
*      dpdAssumption.mean = 10.0;
*      dpdAssumption.use_milan_aaa_ce = 0;
*      dpdAssumption.milan_aaa_ce = 0.0;
*      dpdAssumption.standard_deviation = 1.0;
*      dpdAssumption.discounted_recoveries = 0;
*      std::fill(dpdAssumption.revolving_default_factors, dpdAssumption.
      revolving_default_factors+500, 0.01);
*      dpdAssumption.num_scenarios = 3;
*      dpdAssumption.use_revolving_def_timing = 1;
*      dpdAssumption.rating_cap_primary = 3;
*      dpdAssumption.rating_cap_surveillance = 5;
*
*      assert(0 == set_dpd_assumption(pDeal, &dpdAssumption));
*
*      double currentDefaultTimingVector[] = {0.01};
*      double revolvingDefaultTimingVector[] = {0.02};
*      assert(0 == set_dpd_current_default_timing(pDeal,
      currentDefaultTimingVector, 1, 0));
*      assert(0 == set_dpd_revolving_default_timing(pDeal,
      revolvingDefaultTimingVector, 1, 0));
*
*      assert(0 == set_dpd_el_pd_factors(pDeal, 0.55, 0.45));
*      for (int i = 1; i <= 30; ++i)
*      {
*          assert(0 == set_dpd_threshold(pDeal, "Aaa", i, 0.000005 + 0.00003*i));
*          assert(0 == set_dpd_threshold(pDeal, "Aa1", i, 0.00005 + 0.00003*i));
*          assert(0 == set_dpd_threshold(pDeal, "Aa2", i, 0.00001 + 0.00003*i));
*      }
*
*      assert(0 == run_default_probability_distribution(pDeal));
*
*      int scenariosNum = get_dpd_scenarios(pDeal, 0, 0);
*      assert(scenariosNum > 0);
*      DPD_SCENARIO *scenarios = new DPD_SCENARIO[scenariosNum];
*      assert(scenariosNum == get_dpd_scenarios(pDeal, scenarios, scenariosNum));
*      delete [] scenarios;
*
*      DPD_RESULT result;
*      assert(0 == get_dpd_results(pDeal, "A1", &result));
*
*      const int bondcfIds[] = {FLOW_BOND_BALANCE,
      FLOW_BOND_INTEREST, FLOW_BOND_PRINCIPAL,
      FLOW_BOND_INTEREST_DUE, FLOW_BOND_RATE};
*      const int collcfIds[] = {
*          FLOW_COLLATERAL_BALANCE,
      FLOW_COLLATERAL_INTEREST, FLOW_COLLATERAL_PRINCIPAL,
      FLOW_COLLATERAL_SCHED_PRINCIPAL,
      FLOW_COLLATERAL_PREPAYMENTS,
*          FLOW_COLLATERAL_DEFAULTS,
      FLOW_COLLATERAL_LOSSES, FLOW_COLLATERAL_LIQUIDATIONS,
      FLOW_COLLATERAL_REINVESTMENT,
      FLOW_COLLATERAL_INTEREST_OF_BUYS
*      };
*
*      const double *bondcf[5];
*      const double *collcf[10];
*
*      // get cf for scenario 2
*      for (int i = 0; i < sizeof(collcfIds)/sizeof(collcfIds[0]); ++i)
*      {
*          collcf[i] = get_collateral_flow_sim(pDeal, 2, collcfIds[i]);
*          assert(collcf[i] != NULL);
*      }
*      for (int i = 0; i < sizeof(bondcfIds)/sizeof(bondcfIds[0]); ++i)
*      {
*          bondcf[i] = get_bond_flow_sim(pDeal, 2, "A1", bondcfIds[i]);
*          assert(bondcf[i] != NULL);
*      }
*
*      // get average cf
*      for (int i = 0; i < sizeof(collcfIds)/sizeof(collcfIds[0]); ++i)
*      {
*          collcf[i] = get_collateral_flow_sim(pDeal, 0, collcfIds[i]);
*           assert(collcf[i] != NULL);
*       }
*       for (int i = 0; i < sizeof(bondcfIds)/sizeof(bondcfIds[0]); ++i)
*       {
*           bondcf[i] = get_bond_flow_sim(pDeal, 0, "A2", bondcfIds[i]);
*           assert(bondcf[i] != NULL);
*       }
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.122 int CHASAPI run_FFIEC_test (void ∗ *tid,* int *prepay_type,* double ∗ *prepay_rates*)

Run the deal for the FFIEC test

Since

3.2.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*prepay_type*	The input prepay type for FFIEC test, must be one of prepayment type.
in	*prepay_rates*	The input prepay rate for FFIEC test scenarios, it is a pointer to the prepay rates array whose length must be 7. If input NULL, FFIEC tests run deal with no prepayments.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Note

Current number of FFIEC scenarios is 7,1∼7 means interest rate +300, +200, +100, +0, -100, -200 , -300bps.

Example:

```
*       void* pDeal = NULL;
*       //deal has been opened.
*       run_FFIEC_test(pDeal, PREPAY_CURVE_SMM, NULL);
*
```

22.3.3.123 int CHASAPI run_monte_carlo_simulation (void ∗ *tid*)

This method is used to run the Monte Carlo simulation. After calling this method, users can use methods get_monte-_carlo_result(), get_bond_flow_sim() and get_collateral_flow_sim() to get the corresponding simulation results.

Since

2.1.0

Availability CDOnet, SFW

Precondition

set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	Success.
-1	Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*
*       assert(0 == run_monte_carlo_simulation(pDeal));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.124 int CHASAPI set_balloon_extension_assumptions (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int ∗ *months,* double ∗ *rates,* int *length,* int *delay,* long *loan_num*)

Sets the balloon extension assumption used for the specified piece of collateral.

New feature Subject to change

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	If reremic deal, this is the id, otherwise null
in	*months*	A pointer to an array of number of months for each balloon extension.
in	*rates*	A pointer to an array of extension penalty rates in percentage for each balloon extension.
in	*length*	The number of extensions set, up to 3 extensions
in	*delay*	The number of months delayed as the extension penalties are paid.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid pointer
-3	Error - Invalid loan number
-5	Error - Invalid value passed
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Note

For CMBS only

Example:

```
*       void* pDeal = NULL;
*       //deal has been opened
*
*       int blMonths[]={10,15,25};
*       // rates: 5%, 10%, 15%
*       double blRates[] = {5, 10, 15};
*       int blLength = 3;
*       int blDelay = 10;
*       long loan_num = 94;
*       int ret = set_balloon_extension_assumptions(pDeal,NULL,blMonths,
      blRates,blLength,blDelay,loan_num);
*       if(ret < 0)
*       {
*           //error handling
*       }
*
```

22.3.3.125 int CHASAPI set_bankloan_call_adj_param (void ∗ *tid,* const BANKLOAN_CALL_ADJ_PARAM ∗ *bankloan_adj,* int *length*)

This function will set adjustment table parameter alpha and phi for bank loan call.

Since

3.0.0

Availability CDONet

Precondition

set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bankloan_adj*	The inputs information adjustment table parameter.
in	*length*	The length of BANKLOAN_CALL_ADJ_PARAM.

Return values

0	Success.
-2	Error - Invalid pointer of bank loan adj input struct.
-3	Error - Invalid bank loan adj structure length.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*
*        const int path_number = 100;
*        METRIC_INPUT_STRUCT_EX metric_input_ex;
*        memset(&metric_input_ex, 0, sizeof(METRIC_INPUT_STRUCT_EX));
*        metric_input_ex.shift_amt = 0.005;
*        metric_input_ex.num_paths = path_number;
*        metric_input_ex.oas_mode = ENABLE_ALL;
*        assert(0 == set_metrics_input_ex(pDeal, &metric_input_ex));
*
*        const int rate_size = 200;
*        double **pVal = new double*[path_number];
*        for (int i = 0; i < path_number; ++i)
*        {
*            pVal[i] = new double[rate_size];
*        }
*        //Fill the index rate values for each path
*
*        std::vector<BANKLOAN_CALL_ADJ_PARAM> adjs;
*        adjs.clear();
*        BANKLOAN_CALL_ADJ_PARAM adj_1;
*        memset(&adj_1, 0, sizeof(adj_1));
*        adj_1.moodys_rating = MOODYS_RATING_B3;
*        adj_1.esg_spotspread_rating = ESG_RATING_B;
*        adj_1.alpha = 70;
*        adj_1.phi = 6;
*        adjs.push_back(adj_1);
*        set_bankloan_call_adj_param(pDeal, &adjs.front(), 1);
*
```

22.3.3.126 long CHASAPI set_borrower_benefits_rate (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* short *index,* short *vector,* double ∗ *pval*)

Sets borrower benefit discount rate for SLABS deals

This function allows users to set the discount rate applied to the borrower benefit for SLABS deals.

New feature Subject to change

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	Reremic deal id for a child deal, otherwise null.
in	*index*	The index of the borrower benefit. -1 if the given value/vector will be applied to all borrower benefits.
in	*vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the rates (or rate) to be applied to the specified borrower benefit.

Return values

0	Success
-1	Error - Deal not opened
-3	Error - Invalid index
-4	Error - No value passed or the value of vector is negative
-5	Error - Current deal is not SLABS
-99	Error - For details call get_deal_error_msg()

Note

The rates are expressed as decimals. e.g. 65.8% would be 0.658.

Example:

```
*       void* ptid = NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"WSLT2006-1");
*
*       //open deal
*       open_deal_ex(ptid,&deal);
*
*       short bbIdx(1);
*       short len(0);
*       double dRate(0.5);
*       int iret = set_borrower_benefits_rate(pDeal, NULL, bbIdx, len, &dRate);
*       if(iret < 0)
*           //Error handle
*
*       close_deal_ex(ptid,&deal);
*
```

22.3.3.127 int CHASAPI set_buy_price_override (void ∗ *tid*, short *override_type*, double ∗ *price*, int *size*)

Overrides the buy price assumption for current deal.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*override_type*	Type to indicate what buy price assumptions to use, refer to enum BUY_PRICE_OVERRIDE_TYPE.
in	*price*	A pointer to the vector of price. Need to input when the override_type equal to BUY_PRICE_OVERRIDE_INPUT.
in	*size*	Size of price vector, or 0 if pval points to a constant. Need to input when the override_type equal to BUY_PRICE_OVERRIDE_INPUT.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid pointer of price
-3	Error - Invalid size
-4	Error - Invalid override type
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     double price[] = {100.0, 98.5, 102.5};
*     set_buy_price_override(pDeal,
      BUY_PRICE_OVERRIDE_INPUT, price, 3);
*
```

22.3.3.128 int CHASAPI set_calculation_method (void * *tid*, PREPAY_DEFAULT_CALC_METHOD_TYPE *method_index*, BOOLYAN *set_sup_remic*)

Sets the calculation method for a deal to run. SFW can let user to choose among several calculation methods to calculate default, prepayment and scheduled principal, when projecting cashflows of a deal.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*method_index*	The specified method type, should be enums of PREPAY_DEFAULT_CALC_METHOD_TYPE.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_msg()

Example:

```
*   void* ptid = NULL;
*   //deal has been opened
*
*   int ret = set_calculation_method(ptid,
      SCHED_PRIN_PREPAY_BEFORE_DEFAULT_PPYDEF, false);
*   if(ret != 0)
*   {
*       //error handling
*   }
*
```

Note

method_index must be one of PREPAY_DEFAULT_CALC_METHOD_TYPE. If not called, default method is PREPAY_DEFAULT_BEFORE_SCHED_PRIN_PPYDEF.

Warning

Student loan deal cannot set to JAPANESE_PREPAY_DEFAULT_PPYDEF.

22.3.3.129 int CHASAPI set_call_date_override (void ∗ *tid,* short *override_type,* char ∗ *override_date*)

Overrides the call date assumption for current deal.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*override_type*	Type to indicate what call date assumptions to use, refer to enum CALL_DAT-E_OVERRIDE_TYPE.
in	*override_date*	A pointer to the user input date. Format "YYYYMM" or "YYYYMMDD". Need to input when the override_type equal to CALL_DATE_OVERRIDE_INPUT.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid value or pointer of date
-3	Error - Invalid override type
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     //deal has been opened
*
*     set_call_date_override(pDeal,
      CALL_DATE_OVERRIDE_INPUT, "20160815");
*
```

22.3.3.130 long CHASAPI set_call_option (void ∗ *tid,* short *type,* BOOLYAN *set_sup_remic*)

Sets the call option type for the deal. It is an extension of clean_up_call_ex(). The setting will apply to all underlying deals if set_sup_remic is TRUE. By default, deal will run into maturity instead of call.

New feature Subject to change

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	Type of the call option that the user want to run, should be enums of CALL_O-PTION_TYPE.
in	*set_sup_remic*	The clean-up call setting will apply to all underlying deals if set_up_remic is TRUE. It will NOT apply to underlying deals if it is FALSE.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid call option
-99	Error - Invalid dso identifier (tid) and other errors

Example:

```
    *      void* pDeal = NULL;
    *      CMO_STRUCT *pCmo = new CMO_STRUCT();
    *      memset(pCmo, 0, sizeof(*pCmo));
    *      strcpy(pCmo->dealid, "PARAGONM11");
    *
    *      set_engine_preference(
          PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
    *      assert(0 == open_deal_ex(pDeal, pCmo));
    *
    *      short callType = RUN_TO_MATURITY;
    *      bool isSetUpRemic = false;
    *      assert(0, set_call_option(pDeal, callType, isSetUpRemic));
    *
    *      assert(0 == run_deal_ex(pDeal, pCmo));
    *      assert(0 == close_deal_ex(pDeal, pCmo));
    *      delete pCmo;
    *      pCmo = NULL;
    *
```

See Also

- clean_up_call_ex()

22.3.3.131 int CHASAPI set_call_price_override (void ∗ *tid,* short *override_type,* double ∗ *price,* int *size*)

Overrides the call price assumption for current deal.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*override_type*	Type to indicate what call price assumptions to use, refer to enum CALL_PRICE_OVERRIDE_TYPE.
in	*price*	A pointer to the vector of price. Need to input when the override_type equal to CALL_PRICE_OVERRIDE_INPUT.
in	*size*	Size of price vector, or 0 if pval points to a constant. Need to input when the override_type equal to CALL_PRICE_OVERRIDE_INPUT.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid pointer of price
-3	Error - Invalid size
-4	Error - Invalid override type
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     double price[] = {100.0, 98.5, 102.5};
*     set_call_price_override(pDeal,
      CALL_PRICE_OVERRIDE_INPUT, price, 3);
*
```

22.3.3.132 void CHASAPI set_cdonet_dll_num (const int & *num*)

This method sets the max number of cdonet dll copies in RAM.

Since

2.8.0

Availability CDOnet

Precondition

None.

Parameters

in	*num*	The max cdonet dll number to set.

Return values

None	

Example:

```
*       set_cdonet_dll_num(8);
*
```

Note

- This function should be called before open_deal_ex
- For 32-bit system, the max dll number is capped by 16, and for 64-bit system the max dll number is capped by 4096.

22.3.3.133 void CHASAPI set_cdonet_unload_flag (bool *unload_dll*)

Set the keep dll space flag to determine release dll space in memory or not when closing deals for CDONET.

Since

3.3.0

Availability CDONET

Parameters

in	*unload_dll*	Means release dll space in memory or not when closing deals. True means realse, False means not.

Return values

None.	

Example:

```
*       set_cdonet_unload_flag(true);
*
```

22.3.3.134 int CHASAPI set_cmbs_loan_extension_assumption (void ∗ *tid,* BOOLYAN *use_default,* BOOLYAN *apply_flag,* BOOLYAN *non_perf_loan,* int *maturity_cutoff,* int *extend_years,* double *edf_threshold*)

Set the loan extension assumptions for CMBS deal.

Since

3.1.0

Availability SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*use_default*	On/off switch suggesting if apply CMBS default loan extension assumption , 0 for "No", Non-0 stands for "Yes".
in	*apply_flag*	On/off switch suggesting if apply CMBS loan extension assumption, 0 for "Not Use", Non-0 stands for "Use".
in	*non_perf_loan*	On/off switch suggesting if CMBS loan extension assumption applies to non performing loan ,1 for on and 0 for off.
in	*maturity_cutoff*	Apply the assumption if a loan's maturity is before the cutoff date ,format "YY-YYMMDD".
in	*extend_years*	Number of years to extend.
in	*edf_threshold*	Apply the extention assumption if edf of the loan is over the threshold, should be value between 0 and 1.0.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Not a CMBS deal.
-3	Error - Invalid maturity_cutoff date.
-4	Error - Invalid number of years to extend, should be >= 0.
-5	Error - Invalid threshold, should be between 0 and 1.0.
-99	Error - Other error, use get_deal_error_msg() to see details.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "CMBS_CCC070C3");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      set_service_advances_ex(pDeal,
      SERVICER_ADVANCES_BOTH, false);
*      assert(0 == enable_sfw_delinq_projection(pDeal, false));
*      double edf[5] = {.3, 0, 0, 0, 0};
*      assert(0 == set_loan_edf(pDeal, NULL, -1, edf, 5));
*
*      assert(0 == set_cmbs_loan_extension_assumption(pDeal, false, true,
       true, 20180630, 3, 0.2));
*
*      assert(0 == run_deal_ex(pDeal, pCmo)); // userPd are applied to loan after calling
       run_deal_ex()
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.135 int CHASAPI set_cmm_custom_scenario (void ∗ *tid,* CMM_FACTOR_TYPE *cmm_factor_type,* CMM_FACTOR *factor,* const double ∗ *value,* int *length*)

Sets CMM custom scenario data.

Since

3.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to CMM with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*cmm_factor_-type*	The type of CMM factor type. Must be one of CMM_FACTOR_TYPE.
in	*factor*	The CMM factor. Must be one of CMM_FACTOR.
in	*value*	A pointer to a double array, the array length must be 40.
in	*length*	The length of value, the length must be 40.

Return values

0	Success
-1	Error - Deal not opened
-2	Error - Invalid parameters.
-99	Error - For details call get_deal_error_msg()

Example:

```
*       #include "MarkitCMMProvider/MarkitCMMProvider.h"
*       #pragma comment(lib, "WSACMMProvider.lib")
*
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       int ret = open_deal_ex(pDeal, pCmo);
*       if(ret < 0)
*       {
*           //Error handling
*       }
*
*       set_moodys_credit_model_settings(pDeal,
       MOODYS_CMM_SETTINGS, false);
*       double value[] = {0.056999998,0.056258359,0.054541469,0.056365418,0.064670191,0.0782126,0.086426401,
       0.090757341,0.091721907,0.09174448,0.091839638,0.089244394,0.08650013,0.084163132,0.079800591,0.073180208,0.
       067184458,0.061098261,0.056628218,0.05455205,0.054580388,0.0547897323333333,0.0549990766666667,0.055208421,0
       .055474019,0.055556188,0.055709429,0.055890002,0.05600625,0.055872004,0.055737758,0.055525441,0.055487661,0.
       055262222,0.055346231,0.055127888,0.055127888,0.054981022,0.054845071,0.054779229};
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_REALGDPGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_UNEMPRATE, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_FEDFUNDSRATE, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_TSY10Y, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_CPIINFRATE, value, 40);
*
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_POPGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_NUMHOUSEHOLDSGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_RETAILSALESGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_TOTNONFARMEMPGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_NOMPERSONALINCGROWTH, value, 40);
*
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_HOMEPRICEGROWTH, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_BAACORPYIELD, value, 40);
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_ME,
       CMM_ME_CREPXIDXGROWTH, value, 40);
*
*       set_cmm_custom_scenario(pDeal, CMM_FACTOR_TYPE_IR,
       CMM_IR_LIBOR1M, value, 40);
*
```

```
*       char* custom_scen_name = "scenario1";
*
*       SetupCMMCustomModel(pDeal, "user", "123456");
*       set_current_moodys_cmm_scenario(pDeal, NULL, custom_scen_name);
*       generate_cmm_custom_result_output(pDeal, custom_scen_name);
*
*       run_deal_ex(pDeal,pCmo);
*       close_deal_ex(pDeal,pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

The CMM Macro factors(CMM_ME_REALGDPGROWTH to CMM_ME_CREPXIDXGROWTH) all need to input. The CMM IR factors(CMM_IR_LIBOR1M to CMM_IR_MMOVINGAVGCMT) at least input one factor.

22.3.3.136 long CHASAPI set_credit_card_assump_ex (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* CREDIT_CARD_ASSUMP_TYPE *assump_type,* short *is_vector,* double ∗ *pval,* long *loan_num*)

Sets the constant or vectored yield, repayment, principal payment, default, recovery, and purchase rates to be used for the pool specified by loan_num or for all pools if loan_num = -1, with the ability to apply to underlying deals if a reremic.

Since

1.4.0

Availability SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The underlying deal id or NULL.
in	*assump_type*	Type of credit card rate. Must be one of CREDIT_CARD_ASSUMP_TYPE(def in ccmo.h): • CREDIT_CARD_ASSUMP_YIELD(Portfolio/Annual Yield) • CREDIT_CARD_ASSUMP_REPAYMENT(Repayment Rate) • CREDIT_CARD_ASSUMP_DEFAULT (only use for Markit engine) • CREDIT_CARD_ASSUMP_RECOVERY(Loss Rate) • CREDIT_CARD_ASSUMP_PURCHASE(Purchase Rate) • CREDIT_CARD_ASSUMP_PRINCIPAL_PAYMENT(Principal Payment Rate)
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the prepayment speeds (or speed). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid input parameter
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_msg()

Note

- Yield, repayment, principal payment, default, recovery, and purchase rates are expressed as decimals. 5.25% would be .0525.
- value for current period (0-indexed element) will not be applied.

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      open_deal_ex(tid,pCmo);
*
*      CMO_STRUCT remics[10] = {0};
*      int deal_num = view_reremic_deals(tid,NULL,remics);
*
*      double prepay_rate = 0.28;
*      double purchase_rate = 0.23;
*      double annual_yield = 0.20;
*      double loss_rate = 0.03;
*
*      for(int i = 0; i < deal_num; i++)
*      {
*          set_credit_card_assump_ex(tid, remics[i].dealid,
      CREDIT_CARD_ASSUMP_REPAYMENT, 0, &prepay_rate, -1);
*          set_credit_card_assump_ex(tid, remics[i].dealid,
      CREDIT_CARD_ASSUMP_PURCHASE, 0, &purchase_rate, -1);
*          set_credit_card_assump_ex(tid, remics[i].dealid,
      CREDIT_CARD_ASSUMP_YIELD, 0, &annual_yield, -1);
*          set_credit_card_assump_ex(tid, remics[i].dealid,
      CREDIT_CARD_ASSUMP_RECOVERY, 0, &loss_rate, -1);
*      }
*
*      run_deal_ex(tid,pCmo);
*      close_deal_ex(tid, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.137 int CHASAPI set_current_edf_scenario (void * *tid,* int *idx*)

This function will set a credit scenario for current SEDF module. idx will be index of array scenarios obtains from function get_edf_scenarios().

Since

2.0.1

Availability CDOnet

Precondition

open_deal_ex() has been called.
The current credit model has been set to SEDF with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*idx*	An index indicating which scenario to be set for SEDF.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid idx.
-3	Error - Current credit model is not SEDF.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*        void* pDeal = NULL;
*        CMO_STRUCT *pCmo = new CMO_STRUCT();
*        memset(pCmo, 0, sizeof(*pCmo));
*        strcpy(pCmo->dealid, "1776");
*
*        set_engine_preference(
       PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*        assert(0 == open_deal_ex(pDeal, pCmo));
*
*        assert(0 == set_moodys_credit_model_settings(pDeal,
       MOODYS_SEDF_SETTINGS, false));
*
*        int scenario_count = get_edf_scenarios(pDeal, NULL);
*        assert(scenario_count > 0);
*
*        assert(0 == set_current_edf_scenario(pDeal, scenario_count-1));
*
*        assert(0 == run_deal_ex(pDeal, pCmo));
*
*        assert(0 == close_deal_ex(pDeal, pCmo));
*        delete pCmo;
*        pCmo = NULL;
*
```

See Also

- get_edf_scenarios()
- get_current_edf_scenario()

22.3.3.138 int CHASAPI set_current_moodys_cmm_scenario (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *cmm_scenario*)

Set current CMM scenario of SFW deal.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*cmm_scenario*	A pointer to a CMM scenario string.

Return values

0	Success.
-1	Error - Input cmm_scenario is not exist.
-3	Error - Current credit model is not CMM.
-99	Error - Call get_deal_error_msg() for detail.

See Also

- get_moodys_cmm_scenarios()
- get_current_moodys_cmm_scenario()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "CMBS_BOA00002");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       open_deal_ex(tid, pCmo);
*
*       int scenario_count = get_moodys_cmm_scenarios(tid, NULL, NULL);
*       assert(scenario_count > 0);
*
*       char **scenarios = new char*[scenario_count];
*       for (int i = 0; i < scenario_count; ++i)
*        scenarios[i] = new char[20];
*       assert(scenario_count == get_moodys_cmm_scenarios(tid, NULL, scenarios));
*
*       set_current_moodys_cmm_scenario(tid, NULL, scenarios[3]);
*
*       run_deal_ex(tid, pCmo);
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.139 int CHASAPI set_current_mpa_scenario (void ∗ *tid,* int *idx*)

This function will set a credit scenario for current MPA module. idx will be index of array scenarios obtains from function get_mpa_scenarios().

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*idx*	An index indicating which scenario to be set for MPA.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid idx.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
```

```
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*      assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*
*      int scenario_count = get_mpa_scenarios(pDeal, NULL);
*      assert(scenario_count > 0);
*
*      assert(0 == set_current_mpa_scenario(pDeal, scenario_count-1));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

- get_mpa_scenarios()
- get_current_mpa_scenario()

22.3.3.140 int CHASAPI set_current_pa_scenario (void ∗ *tid,* int *idx*)

This function will set a credit scenario for current PA module. idx will be index of array scenarios obtains from function get_pa_scenarios().

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*idx*	An index indicating which scenario to be set for PA.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid idx.
-3	Error - PA model is not setup.
-99	Error - Call get_deal_error_msg() for detail.

See Also

- get_pa_scenarios()
- get_current_pa_scenario()

Example:

```
*      void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*
*       assert(0 == set_current_pa_scenario(pDeal, 1));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.141 int CHASAPI set_deal_account_default (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, const char ∗ *account_name*, BOOLYAN *account_default*)

Turns the account on and off by setting the account default status.

Since

1.1.0

Availability CDOnet, CHS, SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*account_name*	The name of account to be set.
in	*account_default*	Set value of account default.

Return values

0	SUCCESS
-1	Deal not open
-2	account_name not found
-99	Error, for details call get_deal_error_msg()

Note

This function is only available for Account Type = Liq Fac and Insurance.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       open_deal_ex(tid,pCmo);
*
*       char modRemicDealName[] = "ABF00001";
*       set_deal_account_default(tid, modRemicDealName, "INSURANC-1", true);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.142 int CHASAPI set_deal_fee_override (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* int *fee_id,* short *fee_type,* double *override_value*)

This method override the specific fee of an SFW deal.

Since

2.9.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*fee_id*	The 1-based index of the fee.
in	*fee_type*	The override type of fee, the value is one of MOODYS_FEE_CAL_CODE.
in	*override_value*	The override value of fee.

Return values

0	Success.
-1	Deal not open.
-2	Invalid fee id.
-3	Fee type not found.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*     void *pDeal = NULL;
*     //deal has been opened
*
*     int iret = set_deal_fee_override(pDeal, NULL, 1,
      FEES_TOTAL_BONDS_CALC, 5.0);
*     if (iret < 0)
*          //error handling
*
```

22.3.3.143 int CHASAPI set_deal_hedge_override (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* const char ∗ *hedge_id,* MOODYS_HEDGE_OVERRIDE *hedge_override_info*)

This method override the specific hedge of an SFW deal.

Since

2.9.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*hedge_id*	The id of hedge which need to be overrided.
in	*hedge_override-_info*	The override hedge information.

Return values

0	Success.
-1	Deal not open.
-2	Invalid hedge id or not found the hedge.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*     void *pDeal = NULL;
*     //deal has been opened
*
*     MOODYS_HEDGE_OVERRIDE override_hedge;
*     memset(&override_hedge, 0, sizeof(MOODYS_HEDGE_OVERRIDE));
*     override_hedge.use_paying_margin_override = true;
*     override_hedge.paying_margin_override_from = 20160814;
*     override_hedge.paying_margin_override_to= 20181011;
*     override_hedge.override_paying_margin = 0.08;
*     int iret = set_deal_hedge_override(pDeal, NULL, "FIXED", override_hedge);
*
```

22.3.3.144 int CHASAPI set_default_before_amortization (void * *tid*, BOOLYAN *def_bef_amort*, BOOLYAN *set_sup_remic*)

Sets the timing when default is calculated. If input parameter def_bef_amort is TRUE, default is calculated based on the outstanding balance before amortization.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*def_bef_amort*	If input parameter def_bef_amort is TRUE, default is calculated based on the outstanding balance before amortization.
in	*set_sup_remic*	If TRUE, this setting will be applied to underlying deals; otherwise, it will not.

Return values

0	No error
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     set_default_before_amortization(pDeal, TRUE, TRUE);
*
```

22.3.3.145 int CHASAPI set_default_non_performing_loans (void * *tid,* BOOLYAN *is_defaulted,* short * *non_perf_status,* BOOLYAN *set_sup_remic*)

This method sets non performing loans as default.

Since

2.0.2

Availability SFW CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_defaulted*	Enable treat non performing loans as default or not.
in	*non_perf_status*	An array contains the loan status which will be treaded as default. non_perf_-status size should be 5 at least and CDO deal will ignore this param. possible value of non_perf_status: non_perf_status[NON_PERFORMING_DELINQU-ENT]: -1 to MAX_PERIODS. -1 repesent turn off this flag. The value n>0 represent that treat loan deqlinq n month as non performing loan. non_perf_-status[NON_PERFORMING_BANKRUPTED]: -1 to MAX_PERIODS. -1 repe-sent turn off this flag. The value n>0 represent that treat loan bankrupted n month as non performing loan. non_perf_status[NON_PERFORMING_REO]-:-1,0. -1 represent turn off this flag. 0 represent that treat loan in REO status as non-perfroming loan. non_perf_status[NON_PERFORMING_FORECLOS-ED]:-1,0. -1 repesent turn off this flag. 0 represent that treat loan in foreclose status as non-perfroming loan.
in	*set_sup_remic*	Settings are applied to underlying deals if TRUE. Otherwise, it will not.

Return values

0	Success.
-1	Deal not opened.
-99	Other errors.

Example:

```
*      //Deal has been opened
*
*      short non_perf_status[NON_PERFORMING_SIZE]={-1};
*      non_perf_status[NON_PERFORMING_DELINQUENT]=3;
*       non_perf_status[NON_PERFORMING_BANKRUPTED]=3;
*       non_perf_status[NON_PERFORMING_REO]=0;
*       non_perf_status[NON_PERFORMING_FORECLOSED]=0;
*       int ret = set_default_non_performing_loans(pDeal,true,
      non_perf_status);
*
```

22.3.3.146 long CHASAPI set_default_till_end (void ∗ *tid,* BOOLYAN *val,* BOOLYAN *set_sup_remic*)

Sets assumption default_till_end of Mortgage/ABS, with the ability to apply to underlying deals if it is a reremic.

It sets if default assumption will apply to the end of cash flow projection:

- If val set to TRUE (valid for SFW only), the default assumption will apply till end of projection, which means the defaults will occur to each period till last period of Mortgage/ABS.
- If val set to FALSE, the default assumption will NOT apply to the last N periods of Mortgage/ABS, where N is the recovery lag set by set_recovery_lag_ex(). This is the case if this api is not called.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Note

For CHS engine, if val set to TRUE, error code -99 will return.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*val*	The flag indicates whether default assumption continue till end of payment periods.
in	*set_sup_remic*	If TRUE this will replace any specified underlying deal settings. If FALSE, this will NOT replace any underlying deal settings.

Return values

0	No error
-1	Error: Deal not opened
-99	Error: Invalid dso identifier (tid) or apply True in a CHS deal

Example:

```
*        void* ptid = NULL;
*        CMO_STRUCT deal;
*        memset(&deal, 0, sizeof(CMO_STRUCT));
*        strcpy(deal.dealid,"AL2010-A");
*
*        // open deal
*        open_deal_ex(ptid, &deal);
*
*        double MDR = .0025;
*        set_defaults_ex(ptid, DEFAULT_CURVE_MDR, 0, &MDR, -1, false);
*        // months to liquidation is 3
*        set_recovery_lag_ex(ptid, 3, -1, false);
*
*        // set default till end to true
*        set_default_till_end(ptid, true, false);
*
*        run_deal_ex(ptid,pCmo);
*        close_deal_ex(ptid, &deal);
*
```

See Also

set_recovery_lag_ex()

22.3.3.147 long CHASAPI set_deferment_rates (void ∗ *tid*, short *is_vector*, double ∗ *pval*, long *loan_num*, BOOLYAN *set_sup_remic*)

Sets deferment rates for SFW SLABS deals

Sets the constant or vectored deferment rate for SLABS deals that will be used for the pool specified by loan_num or for all pools if loan_num = -1, with the ability to apply to underlying deals if a reremic.

Since

1.4.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the prepayment speeds (or speed). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Note

- Deferment rates are expressed as decimals. e.g. 6.23% would be .0623.
- Value for current period (0-indexed element) will not be applied.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       open_deal_ex(tid,pCmo);
*
*       short is_vector = 0;
*       double dValue = 0.001;
*       double * pval = &dValue;
*       long loan_num = 1;
*       BOOLYAN set_sup_remic = false;
*       set_deferment_rates(tid, is_vector, pval, loan_num, set_sup_remic);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.148 int CHASAPI set_distressed_property_recovery (void ∗ *tid,* int *loan_number,* DISTRESSED_PROPERTY_RECOVERY ∗ *recovery_inputs*)

Set distressed property recovery information

New feature Subject to change

Since

3.4.0

Availability SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan.
in	*recovery_inputs*	distressed property recovery detail information.

Return values

0	No error
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void *pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*
*      std::vector<WHOLE_LOAN_STRUCT> loans;
*      // set informations for each loan in vector loans
*      set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*
*      set_deal_calc_level(pDeal, CALC_LEVEL_FULL_WITH_LOAN, 1)
      ;
*
*       DISTRESSED_PROPERTY_RECOVERY recovery_inputs;
*       memset(&recovery_inputs, 0, sizeof(recovery_inputs));
*       recovery_inputs.recovery_lag = 6;
*       recovery_inputs.inflation_start_period = 0;
*       recovery_inputs.inflation_rate = 0.02;
*       recovery_inputs.distressed_property_value = 105263;
*       recovery_inputs.variable_foreclosure_cost = 0.01;
*       recovery_inputs.fixed_foreclosure_cost = 1000;
*       set_distressed_property_recovery(pDeal, -1, &recovery_inputs);
*
*      run_deal_ex(pDeal, pCmo);
*
*      // get loan cashflow
*
*      close_deal_ex(pDeal, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.149 int CHASAPI set_dpd_assumption (void ∗ *tid,* const DPD_ASSUMPTION ∗ *assumption*)

This method sets the general assumption for the default distribution to run.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*assumption*	Structure which stores the assumption for the simulation to run.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameter.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*       DPD_ASSUMPTION dpdAssumption;
*       memset(&dpdAssumption, 0, sizeof(dpdAssumption));
*       dpdAssumption.distribution = DPD_DISTRIBUTION_LOGNORMAL;
*       dpdAssumption.scenario_type = 1;
*       dpdAssumption.mean = 10.0;
*       dpdAssumption.use_milan_aaa_ce = 0;
*       dpdAssumption.milan_aaa_ce = 0.0;
*       dpdAssumption.standard_deviation = 1.0;
*       dpdAssumption.discounted_recoveries = 0;
*       std::fill(dpdAssumption.revolving_default_factors, dpdAssumption.
      revolving_default_factors+500, 0.01);
*       dpdAssumption.num_scenarios = 3;
*       dpdAssumption.use_revolving_def_timing = 1;
*       dpdAssumption.rating_cap_primary = 3;
*       dpdAssumption.rating_cap_surveillance = 5;
*
*       assert(0 == set_dpd_assumption(pDeal, &dpdAssumption));
*
*       // add settings for DPD
*
*       assert(0 == run_default_probability_distribution(pDeal));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.150 int CHASAPI set_dpd_current_default_timing (void * *tid,* const double * *timing,* short *size_timing,* BOOLYAN *seasoning*)

This methods sets the default timing curves for current assets.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*timing*	A user allocated array which stores the information of default timing for current assets.
in	*size_timing*	The size of the default timing for current assets.
in	*seasoning*	TRUE if the vector is seasoned and FALSE otherwise.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF00001");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*      double currentDefaultTimingVector[] = {0.01};
*      assert(0 == set_dpd_current_default_timing(pDeal,
      currentDefaultTimingVector, 1, 0));
*
*      // other settings for DPD
*
*      assert(0 == run_default_probability_distribution(pDeal));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.151 int CHASAPI set_dpd_el_pd_factors (void * *tid*, double *el_factor*, double *pd_factor*)

This method sets the E.L. and P.D. table factors.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*el_factor*	E.L. factor.
in	*pd_factor*	D.P. factor.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*       assert(0 == set_dpd_el_pd_factors(pDeal, 0.55, 0.45));
*
*       // other settings for DPD
*
*       assert(0 == run_default_probability_distribution(pDeal));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.152 int CHASAPI set_dpd_revolving_default_timing (void ∗ *tid,* const double ∗ *timing,* short *size_timing,* BOOLYAN *seasoning*)

This methods sets the default timing curves for revolving assets.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*timing*	A user allocated array which stores the information of default timing for revolving assets.
in	*size_timing*	The size of the default timing for revolving assets.
in	*seasoning*	TRUE if the vector is seasoned and FALSE otherwise.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*       double revolvingDefaultTimingVector[] = {0.02};
*       assert(0 == set_dpd_revolving_default_timing(pDeal,
      revolvingDefaultTimingVector, 1, 0));
*
*       // other settings for DPD
*
*       assert(0 == run_default_probability_distribution(pDeal));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.153 int CHASAPI set_dpd_scenarios (void ∗ *tid*, const DPD_SCENARIO ∗ *scenarios*, short *size_scenario*)

This method sets the scenarios for the default distribution to run. This method needs to be called before running the simulation if users choose to use their own scenarios instead of the ones generated by log-normal/inverse normal distributions.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*scenarios*	An array which stores the user-defined scenarios for the simulation to run.
in	*size_scenario*	Size of the user-defined scenarios.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ABF00001");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
     SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*      DPD_SCENARIO userScenarios[] = {
*          {1, 6.00, 8.00},
*          {2, 11.00, 90.00},
*          {3, 13.00, 1.0},
*          {4, 16.00, 1.0}
*      };
*      assert(0 == set_dpd_scenarios(pDeal, userScenarios, sizeof(userScenarios)/sizeof(
     userScenarios[0])));
*
*      // other settings for DPD
*
*      assert(0 == run_default_probability_distribution(pDeal));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.154 int CHASAPI set_dpd_threshold (void ∗ *tid,* const char ∗ *rating,* short *year,* double *threshold*)

This method sets the rating threshold.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*rating*	The rating whose corresponding threshold will be updated.
in	*year*	The year of the rating of which the corresponding threshold will be updated.
in	*threshold*	Threshold value for the specified rating of the specified year.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid parameters.
-3	Error - Current simulation engine is not set to "Default Probability Distribution".
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_DEFAULT_PROBABILITY_DISTRIBUTION));
*
*       for (int i = 1; i <= 30; ++i)
*       {
*           assert(0 == set_dpd_threshold(pDeal, "Aaa", i, 0.000005 + 0.00003*i));
*           assert(0 == set_dpd_threshold(pDeal, "Aa1", i, 0.00005 + 0.00003*i));
*           assert(0 == set_dpd_threshold(pDeal, "Aa2", i, 0.00001 + 0.00003*i));
*       }
*       assert(0 == set_dpd_el_pd_factors(pDeal, 0.55, 0.45));
*
*       // other settings for DPD
*
*       assert(0 == run_default_probability_distribution(pDeal));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

run_default_probability_distribution()

22.3.3.155 long CHASAPI set_draw_rates (void ∗ *tid,* short *type,* short *is_vector,* double ∗ *pval,* long *loan_num,* BOOLYAN *set_sup_remic*)

Sets the draw rates for the specified collateral.

New feature Subject to change

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	The type of draw rate curve. Must be one of DRAW_RATE_TYPE.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the prepayment speeds (or speed).
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-4	Error - Deal type not support draw curve
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Note

Draw rates are expressed as decimals. e.g. 5.25% would be .0525. This method is just for reverse mortgages currently.

Example:

```
*       void* pDeal = NULL;
*       //deal has been opened
*
*       double dRate = 0.08;
*       int ret = set_draw_rates(pDeal, DRAW_CURVE_CPR, 0, &dRate, -1, true);
*       if(ret < 0)
*       {
*           //error handling
*       }
*
*       double dRateVec[5]={0.0,0.05,0.05,0.05,0.1};
*       ret = set_draw_rates(pDeal, DRAW_CURVE_SMM, 5, dRateVec,1, true);
*       if(ret < 0)
*       {
*           //error handling
*       }
*
```

22.3.3.156 int CHASAPI set_edf_default_multiplier (void ∗ *tid*, double *multiplier*)

This method sets default multiplier for SEDF model. By default, SEDF runs as default multiplier with value 1.00.

Since

2.0.1

Availability CDOnet

Precondition

open_deal_ex() has been called.
The current credit model has been set to SEDF with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*multiplier*	multiplier value to be set.

Return values

0	Success.
-1	Deal not open.
-2	Invalid multiplier.
-3	Current credit model is not Stress EDF.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "1776");
*
*       set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_edf_default_multiplier(pDeal, 1.0));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.157 void CHASAPI set_engine_preference (const ENGINE_PREFERENCE & *engine*)

Allows the user to specify the processing engine to use in the event when a CUSIP is supported by both: CHS and SFW engine. It is not thread-specific and should be called in the main thread of the client application. The system default setting is to use SFW engine in the event of an overlap.

Since

0.9.0

Availability ALL

Parameters

in	*engine*	One of the enums from ENGINE_PREFERENCE

Returns

None

Note

Call this function to specify which library to use in the event that a deal (or CUSIP) is supported by both the C-HS and SFW libraries. It is not thread-specific and should be called in the main thread of the client application. The system default setting is to use SFW engine in the event that a deal exists in both the CHS and SFW libraries.

Example:

```
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*
```

22.3.3.158 int CHASAPI set_exchange_rate (void * *tid*, const char * *currency*, double *val*)

Set specified currency exchange rate per global currency.

New feature Subject to change

Since

2.0.0

Availability SFW CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The ISO name of the specified currency.
in	*val*	The exchange rate of the specified currency to the global currency. For example, if the required currency is GBP, the global currency of the deal is USD and pval is 0.6461, then it means that the exchange rate is 0.6461 GBP per USD.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid currency ISO name
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void *pDeal = NULL;
*      // deal is open
*
*      int ret = set_exchange_rate(pDeal, "USD", 1.3247);
*      if (0 != ret)
*      {
*          // error handle
*      }
*
```

22.3.3.159 long CHASAPI set_forbearance_rates (void * *tid*, short *is_vector*, double * *pval*, long *loan_num*, BOOLYAN *set_sup_remic*)

Sets set the SLABS forbearance rate vector.

Sets the constant or vectored forbearance rate for SLABS deals that will be used for the pool specified by loan_num or for all pools if loan_num = -1, with the ability to apply to underlying deals if a reremic.

Since

1.4.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the prepayment speeds (or speed). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Note

- Forbearance rates are expressed as decimals. e.g. 5.25% would be .0525.
- Value for current period (0-indexed element) will not be applied.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       open_deal_ex(tid, pCmo);
*
*       short is_vector = 0;
*       double dValue = 0.001;
*       double * pval = &dValue;
*       long loan_num = 1;
*       BOOLYAN set_sup_remic = false;
*       set_forbearance_rates(tid, is_vector, pval, loan_num, set_sup_remic);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.160 int CHASAPI set_FRA (void ∗ *tid,* const char ∗ *currency,* const char ∗ *rate_type,* short *start_month,* short *end_month,* double *rate_value*)

Set FRA.

Since

3.0.0

Availability SFW, CDOnet, CHS

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The ISO name of the currency of the requested market index.
in	*rate_type*	Type of interest rate. Available inputs: "LIBOR".
in	*start_month*	Number of months until contract effective date.
in	*end_month*	Number of months until contract termination date.
in	*rate_value*	Rate value in decimal.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Currency code not supported
-3	Error - Type of interest rate not supported.
-99	Error - Other error

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT cmos;
*       memset(&cmos, 0, sizeof(cmos));
*       strcpy(cmos.dealid, "CQSCLO2");
*
*       assert(0 == open_deal_ex(tid, &cmos));
*
*       assert(0 == set_FRA(tid, "USD", "LIBOR", 5, 9, 0.012));
*
*       assert(0 == close_deal_ex(tid, &cmos));
*
```

22.3.3.161 int CHASAPI set_global_rates (const char ∗ *currency*, short *rate_size*, short ∗ *rate_types*, double ∗ *rate_values*)

Set market rates in decimal not specific to a deal. Any interest rate set by this function overrides the corresponding interest rate from MWSA rate DB if available.

Since

3.1.0

Availability SFW, CDOnet, CHS

Precondition

None.

Parameters

in	*currency*	The ISO currency code of the specified interest rates (e.g., "USD").
in	*rate_size*	Number of interest rates to be specified.
in	*rate_types*	Array of interest rate types to be specified. Any element in ∗rate_types must be a TREASURY or LIBOR rate and one of: enums of INDEX_TYPE (In indextypes.h) or enums of INDEX_TYPE_EX (SFW and CDOnet deals).
in	*rate_values*	Array of interest rate values corresponding to elements in rate_types.

Return values

0	Success.
-1	Error - Currency code not supported.
-2	Error - At least one element in rate_types is invalid.
-99	Error - Invalid dso identifier (tid) or other errors, please see details by calling get-_deal_error_msg().

Example:

```
*     short rate_types[] = { LIBOR_60, TSY_240, SVR };
*     double rate_values[] = { 0.0217, 0.025, 0.00053 };
*     assert(0 == set_global_rates("USD", 3, rate_types, rate_values));
*
```

22.3.3.162 int CHASAPI set_global_reinvestment (void * *tid*, GLOBAL_REINVESTMENT_INFO *reinv_info*, short *pool_size*, const GLOBAL_REINVESTMENT_ASSET_INFO * *pool_info*)

Sets the global reinvestment setting.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reinv_info*	Global reinvestment info, refer to GLOBAL_REINVESTMENT_INFO.
in	*pool_size*	Number of assets for global reinvestment.
in	*pool_info*	A pointer to the vector of assets for global reinvestment, refer to GLOBAL_R-EINVESTMENT_ASSET_INFO.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid input for pool_size
-3	Error - Invalid pointer for pool_info
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     set_reinvestment_type(pDeal, GLOBAL_REINV);
*
*     GLOBAL_REINVESTMENT_INFO reinv_info;
*     //set values of reinv_info members
*     const int nAssets = 5;
*     std::vector<GLOBAL_REINVESTMENT_ASSET_INFO> assetInfos(nAssets);
*     //set values of each asset members in vecter assetInfos
*
*     set_global_reinvestment(pDeal, reinv_info, nAssets, &assetInfos.front());
*
```

Note

The global reinvestment assets and settings only effect when reinvestment type is set to GLOBAL_REINV by calling set_reinvestment_type.

22.3.3.163 long CHASAPI set_grace_rates (void * *tid*, short *is_vector*, double * *pval*, long *loan_num*, BOOLYAN *set_sup_remic*)

Sets grace rates for SFW SLABS deals

This function allows users to set the percentage of student loans that are in grace period for SFW SLABS deals.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the grace rates (or rate). Value for current period (0-indexed element) will not be applied.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	Success
-1	Error - Deal not opened
-99	Error - For details call get_deal_error_msg()

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       set_engine_preference(
       PICK_CHS_ENGINE_FOR_MAPPED_DEALS);
*       open_deal_ex(tid,pCmo);
*
*       short is_vector = 0;
*       double dValue = 0.001;
*       double * pval = &dValue;
*       long loan_num = 1;
*       BOOLYAN set_sup_remic = false;
*       set_grace_rates(tid, is_vector, pval, loan_num, set_sup_remic);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

- Grace rates are expressed as decimals. e.g. 3.51% would be .0351.
- Value for current period (0-indexed element) will not be applied.

22.3.3.164 long CHASAPI set_index_rate (void ∗ *tid,* const char ∗ *currency,* short ∗ *idx,* short *vector,* double ∗ *pval*)

Sets the constant or vector interest rate that will be used for the specified currency and index.

New feature Subject to change

Since

2.0.0

Availability ALL

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The ISO name of the currency of the requested market index.
in	*idx*	A pointer to the index to set. ∗idx must be one of: • enums of INDEX_TYPE (In indextypes.h). • enums of INDEX_TYPE_EX (SFW and CDOnet deals).
in	*vector*	The length of the vector pointed to by pval, or 0 if pval points to a constant.
in	*pval*	A pointer to the new rate value or values.

Return values

0	No error.
-1	Error - Deal not open.
-2	Error - Invalid currency ISO name.
-4	Error - Invalid index to market index.
-5	Error - Invalid rate pointer.
-6	Error - Currency 'CNY' only support indices from BLR_12 to HPF_60_PLUS, and from XRATE1 to XRATE8.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void *pDeal = NULL;
*      // deal is open
*
*      short index = LIBOR_3;
*      double rate = 0.028;
*      int ret = set_index_rate(pDeal, "USD", &index, 0, &rate);
*      if (0 != ret)
*      {
*          // error handle
*      }
*
```

Note

- The rates are expressed as a decimal: 5.25% would be .0525.
- The required rates for a deal can be determined by calling get_required_index_codes().
- value for current period (0-indexed element) will not be applied.

- Index rates vector apply from the latest update date closest/relative to the settlement date. A floater bond will use period 1 rate assumption at first reset date since deal update date.
- Index rates vector is a monthly rate vector, regardless payment frequency of deal.
- For indices from BLR_12 to HPF_60_PLUS, only effect when currency is 'CNY'.
- Currency 'CNY' only support indices from BLR_12 to HPF_60_PLUS, and from XRATE1 to XRATE8.

22.3.3.165 int CHASAPI set_index_rate_ex (void ∗ *tid*, const char ∗ *currency*, short ∗ *idx*, int *num_paths*, short *rate_size*, double ∗∗ *idx_val*)

Sets simulation path interest rate that will be used for the specified currency and index.

Since

3.0.0

Availability CDOnet, SFW, CHS

Precondition

set_metrics_input_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The ISO name of the currency of the requested market index.
in	*idx*	A pointer to the index to set. ∗idx must be one of: • enums of INDEX_TYPE (In indextypes.h). • enums of INDEX_TYPE_EX (SFW and CDOnet deals).
in	*num_paths*	Number of paths in the user input.
in	*rate_size*	The vector length of each path in two-dimensional array idx_val.
in	*idx_val*	A pointer to a two-dimensional array of interest rates in decimal.

Return values

0	No error.
-1	Error - Deal not open.
-2	Error - Invalid currency ISO name.
-3	Error - Invalid index to market index.
-4	Error - Invalid path number.
-5	Error - Invalid rate vector size of each path.
-6	Error - Invalid rate pointer.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* pDeal = NULL;
*       //deal has been opened.
*
*       const int path_number = 100;
*       METRIC_INPUT_STRUCT_EX metric_input_ex;
*       memset(&metric_input_ex, 0, sizeof(METRIC_INPUT_STRUCT_EX));
*       metric_input_ex.shift_amt = 0.005;
*       metric_input_ex.num_paths = path_number;
*       metric_input_ex.oas_mode = ENABLE_ALL;
*       assert(0 == set_metrics_input_ex(pDeal, &metric_input_ex));
*
```

```
 *      const int rate_size = 200;
 *      double **pVal = new double*[path_number];
 *      for (int i = 0; i < path_number; ++i)
 *      {
 *          pVal[i] = new double[rate_size];
 *      }
 *      //Fill the index rate values for each path
 *
 *      short idx = LIBOR_3;
 *      assert(0 == set_index_rate_ex(pDeal, "USD", &idx, path_number, rate_size, pVal));
 *
 *      assert(0 == close_deal_ex(pDeal, pCmo));
 *      delete pCmo;
 *      pCmo = NULL;
 *
 *      for (int i = 0; i < path_number; ++i)
 *          delete pVal[i];
 *      delete[] pVal;
 *
```

22.3.3.166 int CHASAPI set_indiv_recovery_nonperf (void ∗ *tid*, BOOLYAN *use_indiv_recovery_nonperf*)

This function will set the nonperf assert recovery.

Since

3.0.0

Availability CDONET

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*use_indiv_-recovery_-nonperf*	If use_indiv_recovery_nonperf is set to TRUE, individual recovery rate under each asset will be applied. If use_indiv_recovery_nonperf is set to FALSE, the value of input parameter nonperf_recovery_rate will be applied.

Return values

0	Success.
-1	Error - Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
 *      void* pDeal = NULL;
 *      CMO_STRUCT *pCmo = new CMO_STRUCT();
 *      memset(pCmo, 0, sizeof(*pCmo));
 *      strcpy(pCmo->dealid, "1776");
 *
 *      set_engine_preference(
       PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
 *      assert(0 == open_deal_ex(pDeal, pCmo));
 *
 *      assert(0 == set_indiv_recovery_nonperf(pDeal, false));
 *
 *      assert(0 == close_deal_ex(pDeal, pCmo));
 *      delete pCmo;
 *      pCmo = NULL;
 *
```

22.3.3.167 long CHASAPI set_insurance_coverage (void ∗ *tid*, const char ∗ *issuer*, INSURANCE_CLAIM *type*, short *is_vector*, double ∗ *pval*)

Sets insurance percentage rate

New feature Subject to change

Since

2.4.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*issuer*	The insurer name.
in	*type*	The insurance claim type.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the coverage rate.

Return values

0	Success
-1	Error - Deal not opened.
-2	Error - Invalid pval pointer or pval length.
-3	Error - Invalid insurer.
-4	Error - The value of vector is negative.
-99	Error - For details call get_deal_error_msg().

Note

The rates are expressed as decimals. e.g. 65.8% would be 0.658.

Example:

```
*      void* ptid = NULL;
*      CMO_STRUCT deal;
*      memset(&deal, 0, sizeof(CMO_STRUCT));
*      strcpy(deal.dealid,"SAS059XS");
*
*      //open deal
*      open_deal_ex(ptid,&deal);
*
*      double rate = 0.34;
*      long ret = set_insurance_coverage(ptid, "Ambac Assurance Corporation",
      INSURANCE_CLAIM_COVERAGE, 0, &rate);
*
*      close_deal_ex(ptid,&deal);
*
```

22.3.3.168 int CHASAPI set_int_capital_code_override (void * *tid*, short *int_capital_code_override_type*)

This method gets the implied loss for a bond.

Since

2.5.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*int_capital_code-_override_type*	The interest capitalization code override type to set, should be one of INT_C-APITAL_CODE_OVERRIDE.

Return values

0	Success.
-1	Deal not open.
-2	Invalid override type.
-3	The function is only available for SLABS deals.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "WSLT2006-1");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_int_capital_code_override(pDeal,
     INT_CAPITAL_CODE_OVERRIDE_ANNUALLY));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.169 long CHASAPI set_liquidation_period (void ∗ *tid,* const int *period,* long *loan_num,* BOOLYAN *set_sup_remic*)

Sets the constant liquidation period for SFW deals. The liquidation period is the number of months over which recoveries (from defaulted principal) are realized. The recoveries will be realized in even slices over the number of months indicated. This method can be used to set the liquidation period for an individual loan specified by the loan_num or for all loans if loan_num = -1,with the ability to apply to underlying deals if a reremic.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*period*	The specified liquidation period.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*    void* ptid = NULL;
*    CMO_STRUCT deal;
*    memset(&deal, 0, sizeof(CMO_STRUCT));
*    strcpy(deal.dealid,"AL2010-A");
*
*    // open deal
*    open_deal_ex(ptid,&deal);
*
*    long ret = set_liquidation_period(ptid, 3, 1, false);
*
*    close_deal_ex(ptid,&deal);
*
```

Note

liquidation period should be >= 1.Default settings period=1

See Also

set_liquidation_schedule()

22.3.3.170 int CHASAPI set_liquidation_periodicity (void ∗ *tid,* short *liquidation_periodicity_type,* BOOLYAN *set_sup_remic*)

Sets liquidation periodicity.

Since

2.7.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*liquidation_-periodicity_type*	Type to indicate which liquidation periodicity type to use, refer to enum LIQUI-DATION_PERIODICITY_TYPE.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid liquidation periodicity type.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* ptid = NULL;
*       // deal has been opened
*
*       int ret = set_liquidation_periodicity(ptid,
      LIQUIDATION_QUARTERLY, false);
*       if(ret != 0)
*       {
*           //error handling
*       }
*
```

Note

By default, liquidation periodicity is monthly. The function is not available for SLABS deals.

22.3.3.171 long CHASAPI set_liquidation_schedule (void ∗ *tid,* short *vector_length,* double ∗ *pval,* long *loan_num,* BOOLYAN *set_sup_remic*)

Sets liquidation schedule for SFW deals

Sets the vectored liquidation schedule for SFW deals. The value entered in the schedule for any month is the percent of recoveries/liquidation proceeds that will be realized that month. The schedule must sum to 100%. This method can be used to set the liquidation schedule for an individual loan specified by the loan_num or for all loans if loan_num = -1,with the ability to apply to underlying deals if a reremic.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*vector_length*	The length of liquidation schedule vector.
in	*pval*	A pointer to the liquidation schedule vector.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-3	Error - Invalid loan number
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* ptid = NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"AL2010-A");
*
*       // open deal
*       open_deal_ex(ptid,&deal);
*
*       // double global_schedule[1] ={1};
*       const int vector_len = 4;
*       double schedule_vector[vector_len] = {0,0.5,0,0.5};
*       // long ret = set_liquidation_schedule(ptid, 1, global_schedule, -1, false);
*       long ret = set_liquidation_schedule(ptid, vector_len, schedule_vector, 1,
      false);
*
*       close_deal_ex(ptid,&deal);
*
```

Note

The liquidation schedule are expressed as decimals. e.g. 6.23% would be .0623, liquidation schedule vector max length is 60, and all items sum must equal to 100%.

Warning

The default setting is to use liquidation period. If the user would like to use liquidation schedule instead of liquidation period, then the user must first call this method with loan_num = -1 to set the deal-level liquidation schedule before calling this method to set liquidation schedules for individual loans.

See Also

set_liquidation_period()

22.3.3.172 int CHASAPI set_loan_edf (void ∗ *tid,* const char ∗ *reremic_deal_id_or_null,* long *loan_num,* double ∗ *pd,* int *length*)

This method sets the default probability data for a specified loan.

Since

2.0.1

Availability CDOnet, SFW

Precondition

open_deal_ex() has been called.
For CDONet deal, the current credit model should been set to SEDF with API set_moodys_credit_model_-settings(), but SFW deal does not need.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*loan_num*	Ordinal collateral number in a deal.
in	*pd*	A pointer to an array of double provided by user. At least 5 elements should be provided.
in	*length*	The number of the elements pointed by pd.

Return values

0	Success.
-1	Deal not open.
-2	Invalid loan_num or tid.
-3	Current credit model is not Stress EDF.
-4	Invalid EDF data array.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "1776");
*
*       set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       double userPd[] = {0.1,0.2,0.3,0.4,0.5};
*       assert(0 == set_loan_edf(pDeal, NULL, loan_num, userPd, 5));
*
*       assert(0 == run_deal_ex(pDeal,pCmo)); // userPd are applied to loan after calling
        run_deal_ex()
*
*       double loanEdfValue[5] = {0};
*       assert(5 == get_loan_edf(pDeal, NULL, loan_num, loanEdfValue, 5)); // the value of
        loanEdfValue should be the same as userPd
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

The loan EDF values would be applied after calling run_deal_ex().

22.3.3.173 int CHASAPI set_loan_lgd (void * *tid*, const char * *reremic_deal_id_or_null*, long *loan_num*, double * *lgd*, int *length*)

This method overides the loss given default(LGD) for a specified loan.

Since

2.1.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.
in	*lgd*	A pointer to the user input LGD vector, the max vector length of lgd be overridden is 10.
in	*length*	The length of the user input lgd vector.

Return values

0	Success.
-1	Deal not open.
-2	Invalid loan number ,invalid deal id or invalid length of lgd.
-99	Error - Invalid dso identifier (tid) and other errors, call get_deal_error_msg() for details.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       double userlgd[] = {0.1,0.2,0.3,0.4,0.5};
*       assert(0 == set_loan_lgd(pDeal, NULL, 1, userlgd, 5));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

The loan lgd would be overridden by after calling run_deal_ex().

22.3.3.174 int CHASAPI set_loan_schedule (void ∗ *tid,* long *loan_number,* WHOLE_LOAN_SINK_FUND ∗ *sink_fund_info*)

This function will Enable setting Sinkfund schedule to CDONET bank loan.

Since

3.2.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_number*	The 1-based index of the loan.
in	*sink_fund_info*	Sink fund info structure.

Return values

0	Success.
-1	Error - Set_whole_loan() not called.
-2	Error - Invalid loan number.
-3	Error - Sizes of three vectors are not valid.
-4	Error - Other error.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "STATICLO");
*
*       set_engine_preference(
       PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(tid, pCmo));
*
*       WHOLE_LOAN_SINK_FUND sink_fund_info;
*       sink_fund_info.size = 14;
*       memset(sink_fund_info.pdate, 0, sizeof(int));
*       memset(sink_fund_info.pprin, 0, sizeof(double));
*       int date[]={20150701,20151001,20160101,20160401,20160701,20161001,20170101,20170401,20170701,
       20171001,20180101,20180401,20180701,20181001};
*       double prin[]={0,0,8000,8000,8000,8000,8000,8000,8000,8000,8000,8000,8000,8000};
*       for(int i=0; i < sink_fund_info.size; i++)
*       {
*           sink_fund_info.pdate[i] = date[i];
*           sink_fund_info.pprin[i] = prin[i];
*       }
*       assert(0 == set_loan_schedule(tid, 1, &sink_fund_info));
*
*       assert(0 == run_deal_ex(tid, pCmo));
*       assert(0 == close_deal_ex(tid, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.175 int CHASAPI set_macroeconomic_factor_ex (void ∗ *tid*, const char ∗ *country*, short ∗ *factor_type*, int *num_paths*, short *val_size*, double ∗∗ *factor_val*)

Set the macroeconomic simulation for OAS analysis (where applicable). This function only applies to CMBS using CMM custom scenario for OAS calculation.

Since

3.3.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*country*	The three-character country codes, please refer to ISO 3166-1 alpha-3.(e.g., "USA").
in	*factor_type*	One of the enums from MACROECONOMIC_FACTOR_TYPE.
in	*num_paths*	Number of paths in the user input, it would be the number of rows in two-dimensional array factor_val.
in	*val_size*	The number of columns in two-dimensional array factor_val.
in	*factor_val*	A pointer to a two-dimensional array of quarterly interest rates in decimal.

Return values

0	No error.
-1	Error - Deal not open.
-2	Error - Invalid country ISO name.
-3	Error - Invalid factor_type.
-4	Error - Invalid path number.
-5	Error - Invalid rate vector size of each path.
-6	Error - Invalid rate pointer.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void* pDeal = NULL;
*      //deal has been opened.
*
*      // set path macro economy data
*      double pVal[3][40];
*      double *parray[3] = {pVal[0], pVal[1], pVal[2]};
*      double **pp = parray;
*      for (int i = 0; i < metric_input_ex.num_paths; ++i)
*      {
*          for (int j = 0; j < 40; ++j)
*              pVal[i][j] = 0.004325 + i*0.001 + j*0.00001;
*      }
*      short factorType = REALGDPGROWTH;
*      int iret = set_macroeconomic_factor_ex(pDeal, mdi.
      country, &factorType, metric_input_ex.num_paths, 40, pp);
*
```

Note

If the value of num_paths in set_macroeconomic_factor_ex() is smaller than METRIC_INPUT_STRUCT_E-X::num_paths, the corresponding vector from set_cmm_custom_scenario() will be repeated for the missing paths. If the macro economic factor set for CMM custom scenario, the number of columns in two-dimensional array factor_val must be 40.

22.3.3.176 int CHASAPI set_metrics_input_ex (void ∗ *tid*, METRIC_INPUT_STRUCT_EX ∗ *metric_inputs*)

This function will set metric inputs for deal run, in order to get metric results.

Since

3.0.0

Availability ALL

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*metric_inputs*	The inputs information for market metrics run.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid input simulation number.
-3	Error - Invalid rate shift amount.
-4	Error - Invalid number of paths.
-5	Error - Invalid extended metrics cal mode.
-6	Error - Invalid pointer of metric input struct.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       METRIC_INPUT_STRUCT_EX metric_input_ex;
*       memset(&metric_input_ex, 0, sizeof(METRIC_INPUT_STRUCT_EX));
*       metric_input_ex.shift_amt = 0.00001;
*       metric_input_ex.num_paths = 10;
*       metric_input_ex.oas_mode = ENABLE_ALL;
*       assert(0 == set_metrics_input_ex(pDeal, &metric_input_ex));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.177 void CHASAPI set_missing_exchange_rates_handling (MISSING_EXCHANGE_RATES_HANDLING *handling*)

Sets the handling of missing exchange rates.

New feature Subject to change

Since

2.0.0

Availability ALL

Precondition

None.

Parameters

in	*handling*	The type of handle missing exchange rates, refer to enum MISSING_EXCHA-NGE_RATES_HANDLING.

Return values

void	

Example:

```
*       set_missing_exchange_rates_handling(
      MISSING_EXCHANGE_RATES_TREAT_AS_ERROR);
*
```

22.3.3.178 int CHASAPI set_monte_carlo_assumption (void * *tid*, const MONTE_CARLO_ASSUMPTION * *basic_assumption*, const MONTE_CARLO_DEF_PPY_REC_ASSUMPTION * *def_ppy_rec_assumption*)

This method is used to set the Monte Carlo assumptions.

Since

2.1.0

Availability CDOnet, SFW

Precondition

set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*basic_-assumption*	Structure which contains the basic assumption to run the Monte Carlo simulation.
in	*def_ppy_rec_-assumption*	Structure which contains other required assumption to run the Monte Carlo simulation.

Return values

0	Success.
-1	Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*
*       MONTE_CARLO_ASSUMPTION basic_assumption;
*       // assign members of basic_assumption
*       MONTE_CARLO_DEF_PPY_REC_ASSUMPTION def_ppy_rec_assumption;
*       // assign members of def_ppy_rec_assumption
*       assert(0 == set_monte_carlo_assumption(pDeal, &basic_assumption, &
      def_ppy_rec_assumption));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.179 int CHASAPI set_monte_carlo_correlation (void * *tid,* MONTE_CARLO_CORRELATION_TYPE *type,* char * *field1,* char * *field2,* double *correlation*)

This method sets the correlation between two fields. Depending on the correlation type, the fields can be either issuer names or industry names.

Since

2.1.0

Availability CDOnet

Precondition

set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	Indicates which correlation table the user wants to change, refer to enum MONTE_CARLO_CORRELATION_TYPE.
in	*field1*	Name of field1 whose correlation to field2 will be updated.
in	*field2*	Name of field2 whose correlation to field1 will be updated.
in	*correlation*	The correlation value between field1 and field2.

Return values

0	Success.
-1	Deal not open.
-2	Invalid pointer.
-3	Unrecognized issuer/industry name
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "1776");
*
*      set_engine_preference(
      PICK_CDONET_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*
*      assert(0 == set_monte_carlo_correlation(pDeal,
      MONTE_CARLO_CORRELATION_INDUSTRY, "DJUSST", "DJUSTA", 0.52));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.180 int CHASAPI set_monte_carlo_default_time_and_recovery (void * *tid,* short *num_path,* short *num_loan,* short *default_time,* double *recovery*)

This method sets the default time and recovery rate of a loan for the Monte Carlo simulation.

Since

2.1.0

Availability CDOnet,SFW

Precondition

set_simulation_engine() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*num_path*	Index of the path of the simulation to which the default timing for a specified loan is given.
in	*num_loan*	The loan to be set.
in	*default_time*	-1 means the loan has already defaulted, 1000 means the loan will never default, any other positive integer represents the period in which the loan will default.
in	*recovery*	Recovery rate to be set.

Return values

0	Success.
-1	Deal not open.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*
*       assert(0 == set_monte_carlo_default_time_and_recovery(pDeal
      , 1, 0, 3, 0.2));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.181 long CHASAPI set_moodys_credit_model_settings (void * *tid,* const MOODYS_CREDIT_MODEL_SETTINGS *credit_model,* BOOLYAN *sets_up_only*)

Moody's has five credit models that provide prepayment/default/loss vectors:

- DPLC for RMBS, Autos, SLABS, and Credit Cards (pool-level vectors)
- CMM for CMBS (loan-level vectors)
- MPA for RMBS (loan-level vectors)
- PA for Student Loan, Auto Loan, Lease, Credit Card deal
- Stress EDF for CLOs

Currently DPLC and CMM vectors are delivered as part of a separate data feed. In addition to the vectors provided by the credit models, Moody's also has its own assumptions (e.g., recovery lag, delinquency rate, etc.) For each of the asset classes/credit models. This method allows you to run these Moody's assumptions along with the vectors that you would receive as part of the data feed.

Since

1.4.0

Availability CDOnet, CHS, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*credit_model*	The credit model that will be used in sfw deal. See MOODYS_CREDIT_MODEL_SETTINGS enumerations for details.
in	*sets_up_only*	Flag to indicate whether user-provided vectors will be set. Currently, it used in CMM, SEDF, MPA, PA credit model. For MPA/PA model, sets_up_only dictates whether user-specified index rates will be used. If sets_up_only=1, index rates from MPA/PA model output will be overridden by the user-specified index rates. Index rates from MPA/PA model output will be used if there is no user input of those rates. If sets_up_only=0, index rates from MPA/PA model output will be used. For CMM model, If sets_up_only=1, default and recovery rates from CMM will be overridden by user assumptions. If sets_up_only=0, none of the default and recovery vectors from CMM will be overridden by user assumptions. For SEDF model, If sets_up_only=1, the embedded assumptions on prepayment speeds, recovery rates and recovery lag can be overridden by user assumptions. If sets_up_only=0, none of the SEDF embedded assumptions will be overridden by user assumptions.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid credit model
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_msg()

Example:

```
*       void* ptid = NULL;
*       CMO_STRUCT deal;
*       memset(&deal, 0, sizeof(CMO_STRUCT));
*       strcpy(deal.dealid,"BAFC08R2");
*
*       //open deal
*       open_deal_ex(ptid,&deal);
*
*       long ret = set_moodys_credit_model_settings(ptid,
      MOODYS_CMM_SETTINGS, true));
*
*       close_deal_ex(ptid,&deal);
*
```

22.3.3.182 int CHASAPI set_mpa_analysis_type (void ∗ *tid*, MPA_ANALYSIS_TYPE *type*)

Set the analysis type for MPA model.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	MPA Analysis type.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid MPA analysis type.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*
*       int ret = set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH);
*       if(ret < 0)
*       {
*           //Error handling
*           return;
*       }
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.183 int CHASAPI set_mpa_confidence_level (void ∗ *tid,* double *confidence_level*)

This function is to set confidence level for MPA simulation.

New feature Subject to change

Since

2.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*confidence_level*	Confidence level to be set, the value should be in [0, 1.00].

Return values

0	Success.
-1	Deal not open.
-2	Invalid confidence level, the value should be in [0, 1.00].
-3	Current credit model is not MPA.
-4	Invalid analysis type, this function should be only called with simulation run.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       open_deal_ex(tid, pCmo);
*       set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS, true);
*
*       set_mpa_confidence_level(tid, 1.00);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.184 int CHASAPI set_mpa_custom_scenario (void ∗ *tid,* const char ∗ *factor,* const char ∗ *scope,* const int ∗ *year,* const int ∗ *quarter,* const double ∗ *value,* int *length*)

This function will set a user defined scenario, which contains customized forecast number for several economic indicators.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*factor*	Identifier to set, should be one of factor name. The "Factor And Scope" paragraph shows more detail about this field.
in	*scope*	Scope of the indicator, apply with "US" or the value in table "Region" and "-State".
in	*year*	A pointer to a year array.
in	*quarter*	A pointer to a quarter array.
in	*value*	A pointer to a value array.
in	*length*	The length of year/quarter/value array.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid factor, or invalid scope on economic factor, use get_deal_error_-msg() to see details.
-3	Error - Current credit model is not MPA.
-4	Error - Invalid MPA analysis type for customized scenario.
-99	Error - Call get_deal_error_msg() for detail.

Factor And Scope

Factor Name	Description
"UNEMPLOYMENT"	U.S. Unemployment Rate , apply scope with "US".
"HPI"	U.S. HPI, House Price Index , apply scope with "US".
"GDP"	U.S. HP,Gross Domestic Product , apply scope with "US".
"TSY1Y"	Treasury 1 Year , apply scope with "US".
"TSY10Y"	Treasury 10 Year , apply scope with "US".
"LIBOR6MSPREAD"	6 Month LIBOR Spread , apply scope with "US".
"FREDMAC"	Freddie Mac Rate , apply scope with "US".
"REG_HPI"	Region House Price Index, apply scope with the value in the "Region" table.
"REG_UNEMPLOYMENT"	Region Unemployment Rate, apply scope with the value in the "Region" table.
"STATE_GDP"	State GDP, apply scope with the value in the "State" table.
"TREASURY3M"	Treasury 3 Month, full economy.
"TREASURY6M"	Treasury 6 Month, full economy.
"TREASURY1YR"	Treasury 1 Year, full economy.
"TREASURY5YR"	Treasury 5 Year, full economy.
"TREASURY10YR"	Treasury 10 Year, full economy.
"LIBOR1M"	1 Month LIBOR, full economy.
"LIBOR3M"	3 Month LIBOR, full economy.
"LIBOR6M"	6 Month LIBOR, full economy.
"LIBOR12M"	12 Month LIBOR, full economy.
"PRIME"	Prime, full economy.
"FREDDIEMAC30YR"	Freddie Mac 30 Year, full economy.
"FREDDIEMAC5YR"	Freddie Mac 5 Year, full economy.
"FREDDIEMAC15YR"	Freddie Mac 15 Year, full economy.

State

Scope Name	Description	Scope Name	Description
"AK"	Alaska	"MT"	Montana
"AL"	Alabama	"NC"	North Carolina
"AR"	Arkansas	"ND"	North Dakota
"AZ"	Arizona	"NE"	Nebraska
"CA"	California	"NH"	New Hampshire
"CO"	Colorado	"NJ"	New Jersey
"CT"	Connecticut	"NM"	New Mexico
"DC"	District Of Columbia	"NV"	Nevada
"DE"	Delaware	"NY"	New York
"FL"	Florida	"OH"	Ohio
"GA"	Georgia	"OK"	Oklahoma
"HI"	Hawaii	"OR"	Oregon

"IA"	Iowa	"PA"	Pennsylvania
"ID"	Idaho	"RI"	Rhode Island
"IL"	Illinois	"SC"	South Carolina
"IN"	Indiana	"SD"	South Dakota
"KS"	Kansas	"TN"	Tennessee
"KY"	Kentucky	"TX"	Texas
"LA"	Louisiana	"UT"	Utah
"MA"	Massachusetts	"VA"	Virginia
"MD"	Maryland	"VT"	Vermont
"ME"	Maine	"WA"	Washington
"MI"	Michigan	"WI"	Wisconsin
"MN"	Minnesota	"WV"	West Virginia
"MO"	Missouri	"WY"	Wyoming
"MS"	Mississippi		

Scope Name	Description	Scope Name	Description
"MABI"	Abilene, TX	"MLAN"	Lansing-East Lansing, MI
"MAKR"	Akron, OH	"MLAR"	La Crosse, WI-MN
"MALA"	Albany-Schenectady--Troy, NY	"MLAS"	Las Vegas-Paradise, NV
"MALB"	Albuquerque, NM	"MLAT"	Lawton, OK
"MALE"	Alexandria, LA	"MLEB"	Lebanon, PA
"MALL"	Allentown-Bethlehem--Easton, PA-NJ	"MLET"	Lewiston, ID-WA
"MALN"	Albany, GA	"MLEW"	Lewiston-Auburn, ME
"MALT"	Altoona, PA	"MLEX"	Lexington-Fayette, KY
"MAMA"	Amarillo, TX	"MLIM"	Lima, OH
"MAME"	Ames, IA	"MLIN"	Lincoln, NE
"MANC"	Anchorage, AK	"MLIT"	Little Rock-North Little Rock-Conway, AR
"MAND"	Anderson, IN	"MLOA"	Logan, UT-ID
"MANE"	Anderson, SC	"MLOG"	Longview, TX
"MANI"	Anniston-Oxford, AL	"MLON"	Longview, WA
"MANN"	Ann Arbor, MI	"MLOS"	Los Angeles-Long Beach-Santa Ana, CA
"MAPP"	Appleton, WI	"MLOU"	Louisville-Jefferson County, KY-IN
"MASH"	Asheville, NC	"MLSC"	Las Cruces, NM
"MATA"	Atlantic City-Hammonton, NJ	"MLUB"	Lubbock, TX
"MATH"	Athens-Clarke County, GA	"MLWR"	Lawrence, KS
"MATL"	Atlanta-Sandy Springs-Marietta, GA	"MLYN"	Lynchburg, VA
"MAUB"	Auburn-Opelika, AL	"MMAC"	Macon, GA
"MAUG"	Augusta-Richmond County, GA-SC	"MMAD"	Madison, WI
"MAUS"	Austin-Round Rock-San Marcos, TX	"MMAM"	Mankato-North Mankato, MN

"MBAK"	Bakersfield-Delano, CA	"MMAN"	Manhattan, KS
"MBAL"	Baltimore-Towson, MD	"MMAR"	Madera-Chowchilla, CA
"MBAN"	Bangor, ME	"MMAS"	Mansfield, OH
"MBAR"	Barnstable Town, MA	"MMCA"	McAllen-Edinburg--Mission, TX
"MBAT"	Baton Rouge, LA	"MMCD"	Merced, CA
"MBCR"	Battle Creek, MI	"MMED"	Medford, OR
"MBCY"	Bay City, MI	"MMEM"	Memphis, TN-MS-AR
"MBEA"	Beaumont-Port Arthur, TX	"MMIA"	Miami-Fort Lauderdale-Pompano Beach, FL
"MBEL"	Bellingham, WA	"MMIC"	Michigan City-La Porte, IN
"MBIL"	Billings, MT	"MMID"	Midland, TX
"MBIN"	Binghamton, NY	"MMIL"	Milwaukee-Waukesha--West Allis, WI
"MBIR"	Birmingham-Hoover, AL	"MMIN"	Minneapolis-St. Paul-Bloomington, MN-WI
"MBLC"	Blacksburg--Christiansburg-Radford, VA	"MMIS"	Missoula, MT
"MBLD"	Boulder, CO	"MMNC"	Manchester-Nashua, NH
"MBLM"	Bloomington, IN	"MMOB"	Mobile, AL
"MBLO"	Bloomington-Normal, IL	"MMOD"	Modesto, CA
"MBND"	Bend, OR	"MMOE"	Monroe, MI
"MBOI"	Boise City-Nampa, ID	"MMOG"	Morgantown, WV
"MBOS"	Boston-Cambridge--Quincy, MA-NH	"MMON"	Montgomery, AL
"MBOW"	Bowling Green, KY	"MMOR"	Monroe, LA
"MBRE"	Bremerton-Silverdale, WA	"MMOV"	Mount Vernon-Anacortes, WA
"MBRP"	Bridgeport-Stamford--Norwalk, CT	"MMOW"	Morristown, TN
"MBRW"	Brownsville-Harlingen, TX	"MMUN"	Muncie, IN
"MBSM"	Bismarck, ND	"MMUS"	Muskegon-Norton Shores, MI
"MBSW"	Brunswick, GA	"MMYB"	Myrtle Beach-North Myrtle Beach-Conway, SC
"MBUF"	Buffalo-Niagara Falls, NY	"MNAA"	Napa, CA
"MBUN"	Burlington, NC	"MNAH"	Nashville-Davidson–Murfreesboro–Franklin, TN

"MBUR"	Burlington-South Burlington, VT	"MNAP"	Naples-Marco Island, FL
"MCAJ"	Cape Girardeau-Jackson, MO-IL	"MNEH"	New Haven-Milford, CT
"MCAN"	Canton-Massillon, OH	"MNEO"	New Orleans-Metairie--Kenner, LA
"MCAR"	Carson City, NV	"MNEY"	New York-Northern New Jersey-Long Island, NY-NJ-PA
"MCAS"	Casper, WY	"MNIL"	Niles-Benton Harbor, MI
"MCCF"	Cape Coral-Fort Myers, FL	"MNOW"	Norwich-New London, CT
"MCED"	Cedar Rapids, IA	"MNPT"	North Port-Bradenton--Sarasota, FL
"MCHA"	Champaign-Urbana, IL	"MOCA"	Ocala, FL
"MCHE"	Cheyenne, WY	"MOCE"	Ocean City, NJ
"MCHI"	Chicago-Joliet--Naperville, IL-IN-WI	"MODE"	Odessa, TX
"MCHO"	Chico, CA	"MOGD"	Ogden-Clearfield, UT
"MCHR"	Charlotte-Gastonia--Rock Hill, NC-SC	"MOKL"	Oklahoma City, OK
"MCHS"	Charleston-North Charleston--Summerville, SC	"MOLY"	Olympia, WA
"MCHT"	Chattanooga, TN-GA	"MOMA"	Omaha-Council Bluffs, NE-IA
"MCHV"	Charlottesville, VA	"MORL"	Orlando-Kissimmee--Sanford, FL
"MCHW"	Charleston, WV	"MOSH"	Oshkosh-Neenah, WI
"MCIN"	Cincinnati-Middletown, OH-KY-IN	"MOWE"	Owensboro, KY
"MCLA"	Clarksville, TN-KY	"MOXN"	Oxnard-Thousand Oaks-Ventura, CA
"MCLD"	Cleveland, TN	"MPAL"	Palm Bay-Melbourne--Titusville, FL
"MCLE"	Cleveland-Elyria--Mentor, OH	"MPAN"	Panama City-Lynn Haven-Panama City Beach, FL
"MCOE"	Coeur d'Alene, ID	"MPAR"	Parkersburg-Marietta--Vienna, WV-OH
"MCOL"	Columbus, GA-AL	"MPAS"	Pascagoula, MS
"MCOM"	Columbia, MO	"MPEN"	Pensacola-Ferry Pass-Brent, FL

"MCON"	Columbus, IN	"MPEO"	Peoria, IL
"MCOO"	Colorado Springs, CO	"MPHI"	Philadelphia-Camden--Wilmington, PA-NJ-DE-MD
"MCOR"	Corpus Christi, TX	"MPHO"	Phoenix-Mesa--Glendale, AZ
"MCOS"	Columbia, SC	"MPIN"	Pine Bluff, AR
"MCOU"	Columbus, OH	"MPIS"	Pittsfield, MA
"MCOV"	Corvallis, OR	"MPIT"	Pittsburgh, PA
"MCRE"	Crestview-Fort Walton Beach-Destin, FL	"MPLM"	Palm Coast, FL
"MCSB"	College Station-Bryan, TX	"MPOC"	Pocatello, ID
"MCUM"	Cumberland, MD-WV	"MPOR"	Portland-South Portland-Biddeford, ME
"MDAG"	Dalton, GA	"MPOT"	Portland-Vancouver--Hillsboro, OR-WA
"MDAI"	Danville, IL	"MPOU"	Poughkeepsie--Newburgh-Middletown, NY
"MDAL"	Dallas-Fort Worth-Arlington, TX	"MPRE"	Prescott, AZ
"MDAV"	Davenport-Moline-Rock Island, IA-IL	"MPRO"	Providence-New Bedford-Fall River, RI-MA
"MDAY"	Dayton, OH	"MPRV"	Provo-Orem, UT
"MDEC"	Decatur, IL	"MPSL"	Port St. Lucie, FL
"MDEL"	Deltona-Daytona Beach-Ormond Beach, FL	"MPUE"	Pueblo, CO
"MDEN"	Denver-Aurora--Broomfield, CO	"MPUG"	Punta Gorda, FL
"MDES"	Des Moines-West Des Moines, IA	"MRAC"	Racine, WI
"MDET"	Detroit-Warren-Livonia, MI	"MRAL"	Raleigh-Cary, NC
"MDEZ"	Decatur, AL	"MRAP"	Rapid City, SD
"MDNV"	Danville, VA	"MREA"	Reading, PA
"MDOT"	Dothan, AL	"MRED"	Redding, CA
"MDOV"	Dover, DE	"MREN"	Reno-Sparks, NV
"MDUB"	Dubuque, IA	"MRIC"	Richmond, VA
"MDUL"	Duluth, MN-WI	"MRIV"	Riverside-San Bernardino-Ontario, CA
"MDUR"	Durham-Chapel Hill, NC	"MROA"	Roanoke, VA
"MEAU"	Eau Claire, WI	"MROC"	Rockford, IL
"MELC"	El Centro, CA	"MROE"	Rochester, MN
"MELI"	Elizabethtown, KY	"MROH"	Rochester, NY
"MELK"	Elkhart-Goshen, IN	"MROM"	Rocky Mount, NC

"MELM"	Elmira, NY	"MROR"	Rome, GA
"MELP"	El Paso, TX	"MSAA"	Santa Rosa-Petaluma, CA
"MERI"	Erie, PA	"MSAC"	Sacramento–Arden--Arcade–Roseville, CA
"MEUG"	Eugene-Springfield, OR	"MSAD"	Sandusky, OH
"MEVA"	Evansville, IN-KY	"MSAE"	Salem, OR
"MFAE"	Fayetteville, NC	"MSAF"	San Francisco-Oakland-Fremont, CA
"MFAI"	Fairbanks, AK	"MSAG"	Saginaw-Saginaw Township North, MI
"MFAM"	Farmington, NM	"MSAJ"	San Jose-Sunnyvale-Santa Clara, CA
"MFAR"	Fargo, ND-MN	"MSAL"	Salinas, CA
"MFAY"	Fayetteville-Springdale--Rogers, AR-MO	"MSAN"	San Diego-Carlsbad-San Marcos, CA
"MFLA"	Flagstaff, AZ	"MSAO"	San Angelo, TX
"MFLI"	Flint, MI	"MSAS"	Salisbury, MD
"MFLO"	Florence, SC	"MSAT"	Santa Barbara-Santa Maria-Goleta, CA
"MFLR"	Florence-Muscle Shoals, AL	"MSAU"	St. George, UT
"MFLW"	Fond du Lac, WI	"MSAV"	Savannah, GA
"MFOC"	Fort Collins-Loveland, CO	"MSAX"	Santa Cruz-Watsonville, CA
"MFOR"	Fort Smith, AR-OK	"MSAY"	Salt Lake City, UT
"MFOW"	Fort Wayne, IN	"MSAZ"	San Antonio-New Braunfels, TX
"MFRE"	Fresno, CA	"MSEA"	Seattle-Tacoma--Bellevue, WA
"MGAD"	Gadsden, AL	"MSFE"	Santa Fe, NM
"MGAG"	Gainesville, GA	"MSHB"	Sheboygan, WI
"MGAI"	Gainesville, FL	"MSHE"	Sherman-Denison, TX
"MGLF"	Glens Falls, NY	"MSHR"	Shreveport-Bossier City, LA
"MGOL"	Goldsboro, NC	"MSIO"	Sioux City, IA-NE-SD
"MGRA"	Grand Rapids-Wyoming, MI	"MSIU"	Sioux Falls, SD
"MGRB"	Green Bay, WI	"MSLO"	San Luis Obispo-Paso Robles, CA
"MGRE"	Great Falls, MT	"MSOU"	South Bend-Mishawaka, IN-MI
"MGRF"	Grand Forks, ND-MN	"MSPA"	Spartanburg, SC
"MGRJ"	Grand Junction, CO	"MSPF"	Springfield, OH
"MGRL"	Greeley, CO	"MSPI"	Springfield, MA
"MGRN"	Greensboro-High Point, NC	"MSPM"	Springfield, MO

"MGRV"	Greenville-Mauldin--Easley, SC	"MSPO"	Spokane, WA
"MGUL"	Gulfport-Biloxi, MS	"MSPR"	Springfield, IL
"MGVL"	Greenville, NC	"MSTC"	St. Cloud, MN
"MHAF"	Hanford-Corcoran, CA	"MSTE"	Steubenville-Weirton, OH-WV
"MHAI"	Harrisburg-Carlisle, PA	"MSTG"	State College, PA
"MHAN"	Harrisonburg, VA	"MSTJ"	St. Joseph, MO-KS
"MHAR"	Hartford-West Hartford-East Hartford, CT	"MSTL"	St. Louis, MO-IL
"MHAS"	Hagerstown--Martinsburg, MD-WV	"MSTO"	Stockton, CA
"MHAT"	Hattiesburg, MS	"MSUT"	Sumter, SC
"MHIC"	Hickory-Lenoir--Morganton, NC	"MSWB"	Scranton–Wilkes-Barre, PA
"MHIN"	Hinesville-Fort Stewart, GA	"MSYR"	Syracuse, NY
"MHMT"	Houma-Bayou Cane-Thibodaux, LA	"MTAL"	Tallahassee, FL
"MHOL"	Holland-Grand Haven, MI	"MTAM"	Tampa-St. Petersburg-Clearwater, FL
"MHON"	Honolulu, HI	"MTER"	Terre Haute, IN
"MHOT"	Hot Springs, AR	"MTEX"	Texarkana, TX-Texarkana, AR
"MHOU"	Houston-Sugar Land-Baytown, TX	"MTOL"	Toledo, OH
"MHUN"	Huntsville, AL	"MTOP"	Topeka, KS
"MHUT"	Huntington-Ashland, WV-KY-OH	"MTRE"	Trenton-Ewing, NJ
"MIDA"	Idaho Falls, ID	"MTUC"	Tucson, AZ
"MIND"	Indianapolis-Carmel, IN	"MTUL"	Tulsa, OK
"MIOW"	Iowa City, IA	"MTUS"	Tuscaloosa, AL
"MITH"	Ithaca, NY	"MTYL"	Tyler, TX
"MJAC"	Jacksonville, FL	"MUTI"	Utica-Rome, NY
"MJAK"	Jackson, MI	"MVAD"	Valdosta, GA
"MJAM"	Jackson, MS	"MVAL"	Vallejo-Fairfield, CA
"MJAN"	Janesville, WI	"MVER"	Sebastian-Vero Beach, FL
"MJAS"	Jacksonville, NC	"MVIC"	Victoria, TX
"MJAT"	Jackson, TN	"MVIN"	Vineland-Millville--Bridgeton, NJ
"MJEF"	Jefferson City, MO	"MVIR"	Virginia Beach-Norfolk-Newport News, VA-NC

"MJOB"	Jonesboro, AR	"MVIS"	Visalia-Porterville, CA
"MJOH"	Johnstown, PA	"MWAC"	Waco, TX
"MJON"	Johnson City, TN	"MWAE"	Waterloo-Cedar Falls, IA
"MJOP"	Joplin, MO	"MWAR"	Warner Robins, GA
"MKAK"	Kankakee-Bradley, IL	"MWAS"	Washington-Arlington--Alexandria, DC-VA-MD-WV
"MKAL"	Kalamazoo-Portage, MI	"MWAU"	Wausau, WI
"MKAN"	Kansas City, MO-KS	"MWEN"	Wenatchee-East Wenatchee, WA
"MKIL"	Killeen-Temple-Fort Hood, TX	"MWHE"	Wheeling, WV-OH
"MKIN"	Kingsport-Bristol--Bristol, TN-VA	"MWIC"	Wichita, KS
"MKIS"	Kingston, NY	"MWIH"	Wichita Falls, TX
"MKNE"	Kennewick-Pasco--Richland, WA	"MWII"	Williamsport, PA
"MKNO"	Knoxville, TN	"MWIM"	Wilmington, NC
"MKOK"	Kokomo, IN	"MWIN"	Winchester, VA-WV
"MLAA"	Lafayette, LA	"MWIS"	Winston-Salem, NC
"MLAC"	Lancaster, PA	"MWOR"	Worcester, MA
"MLAD"	Laredo, TX	"MYAK"	Yakima, WA
"MLAE"	Lakeland-Winter Haven, FL	"MYOR"	York-Hanover, PA
"MLAF"	Lafayette, IN	"MYOU"	Youngstown-Warren--Boardman, OH-PA
"MLAH"	Lake Havasu City-Kingman, AZ	"MYUB"	Yuba City, CA
"MLAK"	Lake Charles, LA	"MYUM"	Yuma, AZ

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0, set_mpa_analysis_type(pDeal,
      MPA_CUST_MEDC_SINGLE_PATH));
*
*       int UsUemployment_year[] = {2013,2014,2014,2014,2014,2015,2015,2015,2015,2016,2016,2016,2016,2017,
      2017,2017,2017,2018,2018,2018};
*       int UsUemployment_quarter[] = {4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3};
*       double UsUemployment_value[] = {7.89,7.85,7.77,7.56,7.67,7.42,7.39,7.31,7.26,7.29,7.33,7.28,7.25,7.1
      7,7.21,7.19,7.14,7.03,7.01,6.88};
*       assert(0 == set_mpa_custom_scenario(pDeal, "UNEMPLOYMENT", "US",
      UsUemployment_year, UsUemployment_quarter, UsUemployment_value, 20));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.185 int CHASAPI set_mpa_data_path (const char * *path*)

This function will set location of support data for MPA analysis explicitly, or default location will be used(deal_input-_path/MPA/data/). Note that Support data DB file has been discontinued since MPA version 4.0.45 and the data has been embedded in the MPA API libraries.

Since 2.9.6, this function can also set location of MPA(Mortgage Portfolio Analyzer) API library for MPA analysis explicitly, or default location of MPA(Mortgage Portfolio Analyzer) API library will be same as the application folder for wsa.dll.

Since

2.0.0

Availability SFW

Precondition

NULL.

Parameters

in	*path*	Pointing to the location(folder) of "SupportData.db".

Return values

0	Success.
-99	Error - Call get_deal_error_msg() for detail.

Warning

This function must be called prior to call set_moodys_credit_model_settings() function to set mpa model.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*
*       assert(0 == set_mpa_data_path("./mpadata/"));
*
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.186 int CHASAPI set_mpa_default_loan_data (void * *tid,* const char * *loan_data_field,* const char * *value*)

When running MPA, this function will set default attribute for the loan which missing certain data field.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*loan_data_field*	The name of the loan data field, the "Default Loan Data" paragraph shows more details.
in	*value*	Value for the data field to set.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - invalid parameter loan_data_field or value.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Default Loan Data

loan_data_filed	Default Value	valid input	Note
"Loan ID"	"111000"		
"Mortgage Type"	"Unknown"	"Unknown", "10 yr. CMT balloons","15 yr. CMT balloons","3 yr. Balloon","5 yr. Balloon","7 yr. Balloon","10 yr. Balloon","15 yr. Balloon", "FIXED RATE","Fixed rate simple interest","Fixed rate fully amortizing","E-OMs","Treasury 1/5 caps","Treasury 2/6 caps","Treasury 3/1 yr.","Treasury 5/1 yr.","Treasury 7/1 yr.","Treasury 10/1 yr.","Treasury 10/1 yr (40 yr loans)","Treasury negative amortization", "Treasury no periodic & life caps","Treasury neg am no periodic nor life caps","COFI","COFI 1/5 caps","COFI 2/6 caps, FHLB","COFI negative amortization", "COFI neg am (No caps)","Fixed 5 yrs/COFI neg am","Step loans","6 month LIBOR no neg am","6 month LIBOR neg am","1 mo. LIBOR no neg am", "1 mo. LIBOR neg am","3 mo. LIBOR no neg am","All other neg am","5/25","7/23","7/23 or 5/25","Simple interest","GEM","GPM", "GEM w/ buydown","CD based","ARM Neg Am CMT balloon","Neg Am LIBOR balloon","Neg Am COFI balloon","Fixed 1 yrs/LIBOR","Fixed 2 yrs/LIBOR", "Fixed 3 yrs/LIBOR","Fixed 5 yrs/LIBOR","Fixed 7 yrs/LIBOR","Fixed 7 yrs/COFI no neg am","15 yr LIBOR balloon","1 yr LIBOR","Prime based","Prime based neg am", "ARM 6 Mo. CMT Neg Am","ARM 3 yr CMT Neg Am","ARM 1 yr CMT Neg Am","Arms 1 yr CMT","Arms 3 yr	

"Occupancy Type"	"P: Owner-occupied"	" ","P: Owner-occupied", "S: Second homes","I: Investor", "R: Rentor"	
"Purpose Type"	"P: Purpose Money",	" ","P: Purpose Money", "R: Rate/term refinance", "C: Cash-out refinance", "D: Debt consolidated", "H: Home improvement"	
"Lien Position"	1	Max value is 2	
"Documentation"	"Full Income - No Assets"	"Unknown","Full Income - Full Assets", "Full Assets - Partial Income etc.", "Full Income - No Assets","Partial Income - Stated Assets etc.", "Stated Income - Partial Assets etc.", "No Income - Partial Assets", "No Income - Stated Assets", "No Income - No Assets - VOE", "No Income - No Assets - No VOE"	
"LTV"	80		
"Junior LTV"	20		
"FICO"	670		
"Original Amount"	24000.0		
"Securitized Amount"	20000.0		
"Current Amount"	24000.0		
"Senior Balance"	20000.0		
"HELOC Max Draw Amount"	1.0		1.0 $*$ Initial draw amount
"Gross Coupon"	6.0		
"Gross Margin"	2.5		
"Original Term"	360		
"Amortization Term"	360		
"Cutoff Age"	3		
"Loan Status"	"Current"	" ","Current", "Paid Off", "Delinquent", "Foreclosed", "Bankrupt", "REO","Repurchased", "Liquidated", "Closed",	if set with "Delinquent", usermust input "Delinquent:x", x stands for months in delinquency
"State"	"CA"	"N/A","AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE","FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS","KY", "LA", "ME", "MD", "MA", "MI", "MN", "MS","MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY","NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC","SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV","WI", "WY", "DC", "AB", "BC", "MB", "NB","NL", "NS", "NT", "NU", "ON", "PE", "QC", "SK","YT", "PR"	

"Zip Code"	"94945"		

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*       assert(0 == set_current_mpa_scenario(pDeal, 1));
*
*       assert(0 == set_mpa_default_loan_data(pDeal, "FICO", "500"));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.187 int CHASAPI set_mpa_delinquent_pd (void ∗ *tid,* double *deq_30days,* double *deq_60days*)

This function overrides the probabilities of default for loans with 30/60 days delinquent.

New feature Subject to change

Since

2.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*deq_30days*	Default probability for loans with 30 days delinquent, the value should be in [0.0, 1.0].
in	*deq_60days*	Default probability for loans with 60 days delinquent, the value should be in [0.0, 1.0].

Return values

0	Success.
-1	Deal not open.
-2	Invalid deq_30days or deq_60days value, should be in [0.0, 1.0].

-3	Current credit model is not MPA.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*
*       open_deal_ex(tid, pCmo);
*       set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS, true);
*
*       set_mpa_delinquent_pd(tid, 0.1, 0.1);
*
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.188 int CHASAPI set_mpa_haircut (void ∗ *tid,* short *is_vector,* double ∗ *pval,* BOOLYAN *seasoning*)

This function will set the haircut for MPA analysis with option to apply seasoning or not.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the haircut vector to be set.
in	*seasoning*	Flag for applying seasoning or not.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid haircut vector or invalid seasoning flag.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
```

```
*       assert(0 == set_current_mpa_scenario(pDeal, 1));
*
*       double haircut_vector[500];
*       std::fill(haircut_vector, haircut_vector + 500, 3.0);
*       std::fill(haircut_vector, haircut_vector + 12, 7.0);
*       assert(0 == set_mpa_haircut(pDeal, 500, haircut_vector, 0));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.189 int CHASAPI set_mpa_insurance_non_payment (void * *tid,* double *probability*)

This function will set mpa the probability of non-payment setting.

Since

2.0.1

Availability SFW

Precondition

open_deal_ex() has been called and set MPA mode to simulation.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*probability*	The probability of non-payment, the value should be in [0,1.0].

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid input probability value.
-3	Error - Current credit model is not MPA or not in MPA simulation mode.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*
*       assert(0 == set_mpa_insurance_non_payment(pDeal, 0.5));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.190 int CHASAPI set_mpa_loan_cashflow (void * *tid*, BOOLYAN *enable_loan_cf*)

This function will Enable whether to pass loan level cashflow to WSAAPI or not.

Since

3.1.0

Availability SFW

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*enable_loan_cf*	"True" means enable MPA passing loan cashflow to WSAAP,otherwise not.

Return values

0	Success.
-2	Error - Current calculation level is not CALC_LEVEL_FULL_WITH_LOAN.
-3	Error - set_whole_loan() has not been called.
-4	Error - Current credit model is not MPA.
-5	Error - Other errors.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(pDeal, CALC_LEVEL_FULL_WITH_LOAN, 1)
      ;
*       set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false);
*       set_mpa_analysis_type(pDeal, MPA_MEDC_SINGLE_PATH);
*
*       set_mpa_loan_cashflow(pDeal, true);
*
*       run_deal_ex(pDeal, pCmo);
*
*       close_deal_ex(pDeal, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.191 int CHASAPI set_mpa_mid_course_adj (void * *tid*, BOOLYAN *use*)

This function will enable api to use historical data to generate default/recovery/prepay vectors.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*use*	Flag to indicate whether use historical data to generate performance vectors.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - No need to apply mid course adjustment data as the MPA version is 5.0 or above.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*       assert(0 == set_current_mpa_scenario(pDeal, 1));
*
*       assert(0 == set_mpa_mid_course_adj(pDeal, true));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.192 int CHASAPI set_mpa_multiplier (void ∗ *tid,* MPA_MULTIPLIER_TYPE *type,* short *is_vector,* double ∗ *pval,* long *loan_num*)

This function will set multipliers for MPA analysis, including prepay, default and severity. By default, MPA runs as all multiplier with value 1.00.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

in	*type*	The type of MPA multiplier.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to the multiplier vector to be set.
in	*loan_num*	Loan number, -1 for all collateral loans.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid MPA multiplier or invalid loan number.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ACE06NC1");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*      assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*
*      assert(0 == set_current_mpa_scenario(pDeal, 1));
*      assert(0 == set_mpa_recovery_lag_by_state(pDeal, 30, 35));
*
*      // deal level settings
*      double ppy_multiplier = 2.0;
*      double def_multiplier = 3.0;
*      double sev_multiplier = 4.0;
*      assert(0 == set_mpa_multiplier(pDeal,
      MPA_MULTIPLIER_PREPAY, 0, &ppy_multiplier, -1));
*      assert(0 == set_mpa_multiplier(pDeal,
      MPA_MULTIPLIER_DEFAULT, 0, &def_multiplier, -1));
*      assert(0 == set_mpa_multiplier(pDeal,
      MPA_MULTIPLIER_SEVERITY, 0, &sev_multiplier, -1));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

For deal level input(loan_num = -1), it will use the first value of inputted vectors. For individual loan input, it supports vectors.

22.3.3.193 int CHASAPI set_mpa_offset (void ∗ *tid*, MPA_ANALYSIS_PARAM_OFFSET *type*, int *unit*, double *offset*)

This function will set offset value and offset unit of LTV and FICO when performing MPA analysis in WSAAPI.

New feature Subject to change

Since

2.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	Indicating which field to set(either LTV or FICO), should be one of MPA_ANALYSIS_PARAM_OFFSET.
in	*unit*	0(default value) means disable the offset type; 1 means the offset unit is "%"; 2 means the offset unit is "+/-".
in	*offset*	Offset value for current offset type.

Return values

0	Success.
-1	Deal not open.
-2	Invalid offset type, or unit is not 1 or 2.
-3	Current credit model is not MPA.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
    *      void* tid = NULL;
    *      CMO_STRUCT *pCmo = new CMO_STRUCT();
    *      memset(pCmo, 0, sizeof(*pCmo));
    *      strcpy(pCmo->dealid, "SAS059XS");
    *
    *      open_deal_ex(tid, pCmo);
    *      set_moodys_credit_model_settings(tid,
          MOODYS_MPA_SETTINGS, true);
    *
    *      set_mpa_offset(tid, MPA_ANALYSIS_PARAM_OFFSET_LTV, 1, 2.0
          0);
    *
    *      close_deal_ex(tid, pCmo);
    *      delete pCmo;
    *      pCmo = NULL;
    *
```

22.3.3.194 int CHASAPI set_mpa_optimization (void ∗ *tid,* BOOLYAN *toggle,* double *tail_percent,* double *opt_percent*)

This function will save computation time by only running waterfalls for subset of paths generated by MPA.

- For all paths in the tail section, each of them will be run with waterfall.
- For paths in the non-tails section , only opt_percent ∗ 100% of non-tail paths will be run with waterfall.

New feature Subject to change

Since

2.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*toggle*	1 for open the optimization, 0(default value) for turning off the optimization.
in	*tail_percent*	The tail percentage(in decimals, pass 0.1 indicate 10%) of all path generated.
in	*opt_percent*	The optimization percentage(in decimals, pass 0.1 indicate 10%) for non-tail paths.

Return values

0	Success.
-1	Deal not open.
-2	Invalid tail percentage or optimization percentage input.
-3	Current credit model is not MPA.
-4	Invalid analysis type, this function should be only called with simulation run.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "SAS059XS");
*
*      open_deal_ex(tid, pCmo);
*      set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS, true);
*
*      set_mpa_optimization(tid, true, 0.05, 0.05);
*
*      close_deal_ex(tid, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

By default, API won't use any optimization.

22.3.3.195 int CHASAPI set_mpa_recovery_lag (void ∗ *tid,* short *is_vector,* int ∗ *pval,* long *loan_num*)

This function will set recovery lag on loan level for MPA analysis. User are able to set a constant recovery lag number or vector of recovery lag for MPA analysis.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant.
in	*pval*	A pointer to recovery lag vector or a constant recovery lag value.
in	*loan_num*	Loan number, indicating which loan will be applied by this function, -1 for all collateral loans.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid loan number or pval pointer.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*       assert(0 == set_current_mpa_scenario(pDeal, 1));
*
*       int recoveryLag = 20;
*       assert(0 == set_mpa_recovery_lag(pDeal, 0, &recoveryLag, -1)); // set for all
        loans
*
*       recoveryLag = 10;
*       assert(0 == set_mpa_recovery_lag(pDeal, 0, &recoveryLag, 1));  // loan 1
*
*       int recoveries[200];
*       int *precv = recoveries;
*       std::fill(recoveries, recoveries+sizeof(recoveries)/sizeof(recoveries[0]), 0);
*       std::fill(precv, precv+12, 1); precv+=12;
*       std::fill(precv, precv+12, 2); precv+=12;
*       std::fill(precv, precv+12, 3); precv+=12;
*       std::fill(precv, precv+12, 4); precv+=12;
*       ASSERT_EQ(0, set_mpa_recovery_lag(pDeal, sizeof(recoveries)/sizeof(recoveries[0]
      ), recoveries, 2)); // loan 2
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

For deal level input(loan_num = -1), it will use the first value of inputted vectors. For individual loan input, it supports vectors.

22.3.3.196 int CHASAPI set_mpa_recovery_lag_by_state (void ∗ *tid,* int *judicial_lag,* int *non_judicial_lag*)

This function will set recovery lag (judicial and non-judicial) for MPA analysis.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*judicial_lag*	Input value for judicial lag which applies to all loan in judicial state.
in	*non_judicial_lag*	Input value for non judicial lag which applies to all loan in non-judicial state.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid judicial/non-judicial lags.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ACE06NC1");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*      assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*      assert(0 == set_current_mpa_scenario(pDeal, 1));
*
*      assert(0 == set_mpa_recovery_lag_by_state(pDeal, 20, 25));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.197 int CHASAPI set_mpa_simulation_length (void ∗ *tid*, int *length*)

This function will set length of MPA simulation.

Since

2.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*length*	The MPA simulation length in months. The valid input is [1,499]; if input is 1 or 2, the function will automatically change it to 3, so the return length is [3, 499].

Return values

>0	The simulation length that actually used in MPA analysis.
-1	Error - Deal not open.
-2	Error - Invalid length.
-3	Error - Current credit model is not MPA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_MEDC_SINGLE_PATH));
*       assert(0 == set_current_mpa_scenario(pDeal, 1));
*
*       assert(200, set_mpa_simulation_length(pDeal, 200));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.198 int CHASAPI set_mpa_simulation_path_num (void * *tid,* int *number*)

This function will set the number of path for MPA simulation.

Since

2.0.1

Availability SFW

Precondition

open_deal_ex() has been called and set MPA mode to simulation.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*number*	The number of path for MPA simulation, the value should be in [1, 10000]. The default path number is 10000.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid input simulation number.
-3	Error - Current credit model is not MPA or not in MPA simulation mode.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_LOSS_SIMULATION));
*
*       assert(0 == set_mpa_simulation_path_num(pDeal, 5));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.199 int CHASAPI set_mpa_stress_range (void * *tid,* MPA_ANALYSIS_PARAM *param_type,* double *floor,* double *cap*)

This function will set floor and cap value for Prepay/Default/Severity when performing MPA analysis in WSA API.

New feature Subject to change

Since

2.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*param_type*	Indicating which type to set (either prepay, default, or severity).
in	*floor*	Floor value for current param_type.
in	*cap*	Cap value for current param_type.

Return values

0	Success.
-1	Deal not open.
-2	Invalid floor or cap value.
-3	Current credit model is not MPA.
-99	Other error - Call get_deal_error_msg() for detail.

Example:

```
*      void* tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "SAS059XS");
*
*      open_deal_ex(tid, pCmo);
*      set_moodys_credit_model_settings(tid,
      MOODYS_MPA_SETTINGS, true);
*
*      set_mpa_stress_range(tid, MPA_ANALYSIS_PARAM_PREPAY, 0.
      1, 0.3);
*
*      close_deal_ex(tid, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.200 int CHASAPI set_mpa_thread_count (void ∗ *tid*, int *number*)

This function will set the number of thread for MPA simulation.

Since

3.0.0

Availability SFW

Precondition

open_deal_ex() has been called and set MPA mode to simulation.
The current credit model has been set to MPA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*number*	The number of thread for MPA simulation, the value should be in [1, 32]. The default thread number is equal to CPU core number.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid input thread number.
-3	Error - Current credit model is not in MPA simulation mode.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "ACE06NC1");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
```

```
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_MPA_SETTINGS, false));
*       assert(0 == set_mpa_analysis_type(pDeal,
      MPA_LOSS_SIMULATION));
*
*       assert(0 == set_mpa_thread_count(pDeal, 10));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.201 int CHASAPI set_non_call_end (void ∗ *tid,* int *non_call_end_date*)

This method overrides the date of non-call end with input non_call_end_date; If not called, the date of non-call end will still use the date of non-call end in deal file.

Since

3.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*non_call_end_-date*	This input will be used to override the non_call_end_date, format "YYYYMM-DD".

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid date value
-99	Error - Other error

Example:

```
*       void* tid = NULL;
*       CMO_STRUCT cmos;
*       memset(&cmos, 0, sizeof(cmos));
*       strcpy(cmo.dealid, "CQSCLO2");
*
*       assert(0 == open_deal_ex(tid, &cmos));
*
*       assert(0 == set_non_call_end(tid, 20120328));
*
*       assert(0 == close_deal_ex(tid, &cmos));
*
```

22.3.3.202 int CHASAPI set_non_perf_recovery_lag (void ∗ *tid,* short *value,* BOOLYAN *set_sup_remic*)

This method sets non performing loans months to liquidation.

Since

3.0.0

Availability SFW

Precondition

open_deal_ex() has been called.
set_default_non_performing_loans() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*value*	The lag in months. If value is $<$0, will not be applied to non performing loans.
in	*set_sup_remic*	Settings are applied to underlying deals if TRUE. Otherwise, it will not.

Return values

0	Success.
-1	Deal not opened.
-99	Other errors.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "WSLT2006-1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       short non_perf_status[NON_PERFORMING_SIZE]={-1};
*       non_perf_status[NON_PERFORMING_DELINQUENT]=3;
*       non_perf_status[NON_PERFORMING_BANKRUPTED]=3;
*       non_perf_status[NON_PERFORMING_REO]=0;
*       non_perf_status[NON_PERFORMING_FORECLOSED]=0;
*       assert(0 == set_default_non_performing_loans(pDeal, true,
      non_perf_status, false));
*
*       assert(0 == set_non_perf_recovery_lag(pDeal,4, false));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.203 int CHASAPI set_pa_analysis_type (void * *tid*, PA_ANALYSIS_TYPE *type*)

Set the analysis type for PA model.

Since

2.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	PA Analysis type.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid PA analysis type.
-3	Error - PA model is not setup.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*
*       assert(0 == set_pa_analysis_type(pDeal,
      PA_MEDC_SINGLE_PATH));
*
*       assert(0 == set_current_pa_scenario(pDeal, 1));
*
*       assert(0 == run_deal_ex(pDeal, pCmo));
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.204 int CHASAPI set_pa_custom_scenario (void ∗ *tid,* const char ∗ *factor,* const int ∗ *year,* const int ∗ *quarter,* const double ∗ *value,* int *length*)

This function will set a user defined scenario, which contains customized forecast number for several economic indicators.

Since

2.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*factor*	Identifier to set, should be one of factor name.
in	*year*	A pointer to a year array.

in	*quarter*	A pointer to a quarter array.
in	*value*	A pointer to a value array.
in	*length*	The length of year/quarter/value array.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid PA analysis type for customized scenario.
-3	Error - Current credit model is not PA.
-4	Error - Invalid year,quarter,value pointer or length.
-5	Error - Invalid factor, use get_deal_error_msg() to see details.
-99	Error - Call get_deal_error_msg() for detail.

Factor

Factor Name	Description
"UNEMPLOYMENT"	U.S. Unemployment Rate.
"HPI"	U.S. HPI, House Price Index.
"GDP"	U.S. HP,Gross Domestic Product.
"TSY1Y"	Treasury 1 Year.
"TSY10Y"	Treasury 10 Year.
"LIBOR6MSPREAD"	6 Month LIBOR Spread.
"TREASURY3M"	Treasury 3 Month, full economy.
"TREASURY6M"	Treasury 6 Month, full economy.
"TREASURY1YR"	Treasury 1 Year, full economy.
"TREASURY5YR"	Treasury 5 Year, full economy.
"TREASURY10YR"	Treasury 10 Year, full economy.
"LIBOR1M"	1 Month LIBOR, full economy.
"LIBOR3M"	3 Month LIBOR, full economy.
"LIBOR6M"	6 Month LIBOR, full economy.
"LIBOR12M"	12 Month LIBOR, full economy.
"PRIME"	Prime, full economy.
"FREDDIEMAC30YR"	Freddie Mac 30 Year, full economy.
"NRI"	full economy.
"PCI"	full economy.
"FREDDIEMAC15YR"	full economy.
"FREDDIEMAC1YR"	full economy.
"USEDCARRATE"	full economy.
"NEWCARRATE"	full economy.
"OILPRICE"	full economy.
"USEDCARINDEX"	full economy.
"FREDDIEMAC5YR"	full economy.
"NEWVEHICLESALES"	full economy since PA V1957.
"CPPIAPT"	full economy since PA V1957.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
     MOODYS_PA_SETTINGS, false));
*      assert(0 == set_pa_analysis_type(pDeal,
     PA_CUST_MEDC_SINGLE_PATH));
*
*      int year[]={2013,2014,2014,2014,2014,2015,2015,2015,2015,2016,2016,2016,2016,2017,2017,2017,2017,
     2018,2018,2018};
*      int quarter[]={4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3};
*      double unemploy[] = {7.89,7.85,7.77,7.56,7.67,7.42,7.39,7.31,7.26,7.29,7.33,7.28,7.25,7.17,7.21,7.19
```

```
        ,7.14,7.03,7.01,6.88};
 *       double hpi[] = {2.34,2.98,3.59,4.12,4.46,4.02,4.13,4.19,4.35,4.67,4.95,4.97,5.23,5.69,6.19,8.43,10.1
        3,13.11,14.01,18.63};
 *       double gdp[] = {0.99,1.34,1.77,1.98,2.09,2.43,2.53,2.69,3.16,3.87,4.05,5.11,6.19,9.67,13.42,15.55,16
        .95,18.34,23.39,29.66};
 *       double try1y[] = {0.23,0.27,0.39,0.46,0.55,0.69,0.79,0.92,1.23,1.34,1.56,1.77,1.95,2.03,2.15,2.34,2.
        39,2.41,2.68,2.71};
 *       double try10y[] = {2.25,2.34,2.39,2.44,2.59,2.48,2.41,2.37,2.43,2.57,2.64,2.83,2.96,3.24,3.35,3.81,3
        .49,3.67,4.22,4.53};
 *       double libor6m[] = {0.62,0.61,0.63,0.78,0.96,1.26,1.35,1.67,1.69,1.75,1.86,1.92,2.11,2.23,2.42,2.53,
        2.67,2.92,3.32,3.36};
 *
 *       assert(0 == set_pa_custom_scenario(pDeal,"UNEMPLOYMENT",year,quarter,unemploy,
        20));
 *       assert(0 == set_pa_custom_scenario(pDeal,"HPI",year,quarter,hpi,20));
 *       assert(0 == set_pa_custom_scenario(pDeal,"GDP",year,quarter,gdp,20));
 *       assert(0 == set_pa_custom_scenario(pDeal,"TSY1Y",year,quarter,try1y,20));
 *       assert(0 == set_pa_custom_scenario(pDeal,"TSY10Y",year,quarter,try10y,20));
 *       assert(0 == set_pa_custom_scenario(pDeal,"LIBOR6MSPREAD",year,quarter,libor6m,
        20));
 *
 *       assert(0 == run_deal_ex(pDeal, pCmo));
 *       assert(0 == close_deal_ex(pDeal, pCmo));
 *       delete pCmo;
 *       pCmo = NULL;
 *
```

22.3.3.205 int CHASAPI set_pa_custom_scenario_ex (void ∗ *tid,* const char ∗ *factor,* const char ∗ *country,* const char ∗ *region,* const int ∗ *year,* const int ∗ *quarter,* const double ∗ *value,* int *length*)

This function will set a user defined scenario, which contains customized forecast number for several economic indicators.

Since

2.1.0

Availability SFW,CHS

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*factor*	Identifier to set, should be one of factor name.
in	*country*	country identifier, should be one of country identifier.
in	*region*	Reserve for future use, currently input "" or Null.
in	*year*	A pointer to a year array.
in	*quarter*	A pointer to a quarter array.
in	*value*	A pointer to a value array.
in	*length*	The length of year/quarter/value array.

Return values

0	Success.
-1	Error - Deal not opened.
-2	Error - Invalid PA analysis type for customized scenario.

-3	Error - Current credit model is not PA.
-4	Error - Invalid year,quarter,value pointer or length.
-5	Error - Invalid factor, use get_deal_error_msg() to see details.
-6	Error - Invalid country.
-99	Error - Call get_deal_error_msg() for detail.

Factor

Factor Name	**Description**
"UNEMPLOYMENT"	Unemployment Rate.
"HPI"	HPI, House Price Index.
"GDP"	HP,Gross Domestic Product.
"TSY10Y"	Treasury 10 Year.
"TREASURY3M"	Treasury 3 Month, full economy.
"TREASURY6M"	Treasury 6 Month, full economy.
"TREASURY1YR"	Treasury 1 Year, full economy.
"TREASURY5YR"	Treasury 5 Year, full economy.
"TREASURY10YR"	Treasury 10 Year, full economy.
"LIBOR1M"	1 Month LIBOR, full economy.
"LIBOR3M"	3 Month LIBOR, full economy.
"LIBOR6M"	6 Month LIBOR, full economy.
"LIBOR12M"	12 Month LIBOR, full economy.
"PRIME"	Prime, full economy.
"FREDDIEMAC30YR"	Freddie Mac 30 Year, full economy.
"NRI"	full economy.
"PCI"	full economy.
"FREDDIEMAC15YR"	full economy.
"FREDDIEMAC1YR"	full economy.
"USEDCARRATE"	full economy.
"NEWCARRATE"	full economy.
"OILPRICE"	full economy.
"USEDCARINDEX"	full economy.
"FREDDIEMAC5YR"	full economy.
"NEWVEHICLESALES"	full economy since PA V1957.
"CPPIAPT"	full economy since PA V1957.

country

Country Identifier	**Country**
"US"	US
"IAUS"	Australia
"IDEU"	Germany
"IESP"	Spain
"IFRA"	France
"IIRL"	Ireland
"INLD"	Netherlands
"IPRT"	Portugal
"IITA"	Italy
"IGBR"	UK

Example:

```
*       void* pDeal = NULL;
*       //deal has been opened and set PA model
*
*       int year[]={2013,2014,2014,2014,2014,2015,2015,2015,2015,2016,2016,2016,2016,2017,2017,2017,2017,
       2018,2018,2018};
*       int quarter[]={4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3};
*       double unemploy[] = {7.89,7.85,7.77,7.56,7.67,7.42,7.39,7.31,7.26,7.29,7.33,7.28,7.25,7.17,7.21,7.19
       ,7.14,7.03,7.01,6.88};
*       double hpi[] = {2.34,2.98,3.59,4.12,4.46,4.02,4.13,4.19,4.35,4.67,4.95,4.97,5.23,5.69,6.19,8.43,10.1
       3,13.11,14.01,18.63};
*       double gdp[] = {0.99,1.34,1.77,1.98,2.09,2.43,2.53,2.69,3.16,3.87,4.05,5.11,6.19,9.67,13.42,15.55,16
       .95,18.34,23.39,29.66};
*       double try10y[] = {2.25,2.34,2.39,2.44,2.59,2.48,2.41,2.37,2.43,2.57,2.64,2.83,2.96,3.24,3.35,3.81,3
       .49,3.67,4.22,4.53};
```

```
*
*       assert(0 == set_pa_analysis_type(pDeal,
       PA_CUST_MEDC_SINGLE_PATH));
*       assert(0 == set_pa_custom_scenario_ex(pDeal,"UNEMPLOYMENT","IAUS","",year,
       quarter,unemploy,20));
*       assert(0 == set_pa_custom_scenario_ex(pDeal,"HPI","IAUS","",year,quarter,
       hpi,20));
*       assert(0 == set_pa_custom_scenario_ex(pDeal,"GDP","IFRA","",year,quarter,
       gdp,20));
*       assert(0 == set_pa_custom_scenario_ex(pDeal,"TSY10Y","IITA","",year,quarter
       ,try10y,20));
*       assert(0 == set_pa_custom_scenario_ex(pDeal,"TSY10Y","IGBR","",year,quarter
       ,try10y,20));
*
```

22.3.3.206 int CHASAPI set_pa_data_path (const char ∗ *path*)

This function will set location of support data for PA analysis explicitly, or default location will be used(deal_input_path/PA/data/). Note that Support data DB file has been discontinued since PA version 1.9.45 and the data has been embedded in the PA API libraries.

Since 2.9.6, this function can also set location of PA(Portfolio Analyzer) API library for PA analysis explicitly, or default location of PA(Portfolio Analyzer) API library will be same as the application folder for wsa.dll.

Since

2.0.0

Availability CHS, SFW

Precondition

NULL.

Parameters

in	*path*	Pointing to the location(folder) of "SupportData.db".

Return values

0	Success.
-99	Error - Call get_deal_error_msg() for detail.

Warning

This function must be called prior to call set_moodys_credit_model_settings() function to set pa model.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "AMEXCAMT");
*       set_engine_preference(
       PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*
*       assert(0 == set_pa_data_path("./padata/"));
*
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       assert(0 == set_moodys_credit_model_settings(pDeal,
       MOODYS_PA_SETTINGS, false));
*
*       assert(0 == set_pa_analysis_type(pDeal,
       PA_MEDC_SINGLE_PATH));
*       assert(0 == run_deal_ex(pDeal, pCmo));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.207 int CHASAPI set_pa_default_pool_data (void ∗ *tid*, const char ∗ *paraName*, const char ∗ *value*)

set the default pool data and other settings for PA model.

Since

2.0.0

Availability CHS, SFW

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*paraName*	The name of the parameter:
		• "WALoanAge"
		• "WARemainingTerm"
		• "WAFICO"
		• "WACoupon"
		• "WAPrepayPenaltyTerm"
		• "AverageOriginalLoanAmount"
		• "WAOriginalLTV"
		• "PurposePurchase"
		• "PurposeRefi"
		• "OccupancyOwner"
		• "OccupancySecondHome"
		• "OccupancyInvestor"
		• "Property1Unit"
		• "Property24Unit"
		• "OriginatorThirdParty"
		• "OriginatorRetail"
		• "HARP1"
		• "HARP2"
		• "FHA"
		• "WACAtIssuance"
		• "WAFixedRatePeriod"
		• "ArmIndex"
		• "WAResetInterval"
		• "WALifetimeCap"
		• "WALifetimeFloor"
		• "WAPeriodicCap"
		• "WAInitialCap"
		• "WAMargin"
		• "HistoricalPrepaymentPeriod"
		• "HistoricalPrepaymentRate"
		• "Factor"
		• "Rate30"
		• "Rate60"
		• "Rate90"
		• "CPR"
		• "CDR"
		• "Severity"

Return values

0	Success.
-1	Deal not opened.
-2	Parameter is NULL.
-3	PA model is not setup.
-4	Setting is not supported.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*      assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*
*      assert(0 == set_pa_default_pool_data(pDeal,"WAFICO","600.0");
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.208 int CHASAPI set_pa_multiplier (void ∗ *tid,* PA_MULTIPLIER_TYPE *type,* short *is_vector,* double ∗ *pval,* long *pool_num*)

This function will set multipliers for PA analysis, including prepay, default and severity. By default, PA runs as all multiplier with value 1.00.

Since

3.0.0

Availability SFW, CHS

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	The type of PA multiplier.
in	*is_vector*	The length of the vector pointed to by pval or 0 if pval is a constant. Currently only constant supported.
in	*pval*	A pointer to the multiplier vector to be set.
in	*pool_num*	Pool number, -1 for all collateral pools. Currently only deal level setting supported.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid PA multiplier or invalid pool number or invalid pval.
-3	Error - Current credit model is not PA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*      assert(0 == set_pa_analysis_type(pDeal,
      PA_MEDC_SINGLE_PATH));
*      assert(0 == set_current_pa_scenario(pDeal, 1));
*
*      // deal level settings
*      double ppy_multiplier = 2.0;
*      double def_multiplier = 3.0;
*      double sev_multiplier = 4.0;
*      assert(0 == set_pa_multiplier(pDeal,
      PA_MULTIPLIER_PREPAY, 0, &ppy_multiplier, -1));
*      assert(0 == set_pa_multiplier(pDeal,
      PA_MULTIPLIER_DEFAULT, 0, &def_multiplier, -1));
*      assert(0 == set_pa_multiplier(pDeal,
      PA_MULTIPLIER_SEVERITY, 0, &sev_multiplier, -1));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.209 int CHASAPI set_pa_simulation_path_num (void ∗ *tid,* int *number*)

This function will set the number of path for PA simulation.

Since

3.0.0

Availability SFW, CHS

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*number*	The number of path for PA simulation, the value should be in [1, 10000]. The default path number is 10000.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid input simulation number.
-3	Error - Current credit model is not PA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
      MOODYS_PA_SETTINGS, false));
*      assert(0 == set_pa_analysis_type(pDeal,
      PA_LOSS_SIMULATION));
*
*      assert(0 == set_pa_simulation_path_num(pDeal, 5));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.210 int CHASAPI set_pa_thread_count (void * *tid*, int *number*)

This function will set the number of thread for PA simulation.

Since

3.0.0

Availability SFW, CHS

Precondition

open_deal_ex() has been called.
The current credit model has been set to PA with API set_moodys_credit_model_settings().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*number*	The number of thread for PA simulation, the value should be in [1, 32]. The default path number is equal to CPU core number.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid input thread number.
-3	Error - Current credit model is not PA.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*      void* pDeal = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "AMEXCAMT");
*
*      set_engine_preference(
     PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*      assert(0 == open_deal_ex(pDeal, pCmo));
*
*      assert(0 == set_moodys_credit_model_settings(pDeal,
     MOODYS_PA_SETTINGS, false));
*      assert(0 == set_pa_analysis_type(pDeal,
     PA_LOSS_SIMULATION));
*
*      assert(0 == set_pa_thread_count(pDeal, 5));
*
*      assert(0 == run_deal_ex(pDeal, pCmo));
*
*      assert(0 == close_deal_ex(pDeal, pCmo));
*      delete pCmo;
*      pCmo = NULL;
*
```

22.3.3.211 int CHASAPI set_ppydef_only_on_paydate (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, BOOLYAN *only_on_paydate*)

Sets Prepays only on Pay Dates. Sets flag to indicate whether a loan can only prepay or default on a payment date.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*only_on_paydate*	If TRUE, then collateral only prepays or defaults on the asset payment dates. If FALSE, then collateral can prepay and default on non-payment dates.

Return values

0	SUCCESS
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) and other errors

Example:

```
*    void* ptid = NULL;
*    //deal has been opened
*
*    int ret = set_ppydef_only_on_paydate(ptid, null, TRUE);
*    if(ret != 0)
*    {
*        //error handling
*    }
*
```

Note

If this method is not called, then it will be set to TRUE by default (i.e. loans can only prepay and default on payment dates).

22.3.3.212 int CHASAPI set_prepay_default_compounding_method (void * *tid,* PREPAY_DEFAULT_COMPOUNDING_-METHOD *prepay_default_compound*)

Sets how the prepayment would be compounded, based on "monthly" or asset's "periodicity". By default it is monthly.

Since

4.0.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*prepay_default_-compound*	The specified method of prepayment compounding base, should be enums of PREPAY_DEFAULT_COMPOUNDING_METHOD.

Return values

0	No error.
-1	Error - Deal not opened.
-2	Error - Invalid prepay compounding method.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg().

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       assert(0 == set_prepay_default_compounding_method(pDeal,
      PREPAY_DEFAULT_COMPOUNDING_MONTHLY));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.213 int CHASAPI set_prospectus_prepayment_curves (void * *tid,* short *PPC_index,* int *loan_num,* BOOLYAN *set_sup_remic*)

Sets prepayment speed from prospectus. It will be used for all collateral with ability to apply to underlying deals if it is a reremic.

Since

3.7

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*PPC_index*	The 0-based index of the prospectus_prepayment_curves.
in	*loan_num*	The 0-based index of the loan or -1 to apply to all collateral in the deal. Only -1 is supported.
in	*set_sup_remic*	If TRUE this will replace any specified underlying deal settings. If FALSE, this will NOT replace any underlying deal settings.

Return values

0	Success.
-1	Error - Deal not open.
-2	Error - Invalid PPC index.
-3	Error - Invalid loan number.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT;
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "NL2017-B");
*
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       int num_curves = 0;
*       assert(0 == get_prospectus_prepayment_curves(pDeal, NULL, NULL, 0, &
       num_curves));
*       assert(0 < num_curves);
*
*       assert(0 == set_prospectus_prepayment_curves(pDeal, 0, -1, FALSE));
*
```

22.3.3.214 int CHASAPI set_pv_reinvest_override (void ∗ *tid,* const char ∗ *bondid,* short *override_type*)

Sets the reinvestment override value of a pv test for the specified tranche.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*bondid*	Name of the tranche of which the reinvestment override type is requested.
in	*override_type*	The override types, refer to enum REINV_OVERRIDE_TYPE.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - The specified tranche does not have a PV test
-3	Error - The specified tranche does not exist
-4	Error - Invalid override type
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     set_pv_reinvest_override(pDeal, "A1",
      REINV_OVERRIDE_REINV_PER);
*
```

22.3.3.215 int CHASAPI set_rate_shift_setting (void * *tid*, RATE_SHIFT_SETTING *rate_shift*)

Set the ECON rate shift setting.

Since

3.3.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*rate_shift*	Econ rate shifts settings.

Return values

0	No error.
-1	Error - Deal not open.
-2	Error - Apply CDOnet deals only.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*      void* pDeal = NULL;
*      // deal has been opened.
*
*      // set rate shift setting
*      RATE_SHIFT_SETTING rate_shift;
*      rate_shift.alwaysUseScenRateShift = SCENARIO_NO;
*      rate_shift.rateShiftFromSettle = true;
*      rate_shift.shiftRelativeToCurrentRates = true;
*      set_rate_shift_setting(pDeal, rate_shift);
*
```

22.3.3.216 int CHASAPI set_realized_losses_at_liquidation (void * *tid*, BOOLYAN *realized_at_liquidation*, int *months_prior_liquidation*, BOOLYAN *set_sup_remic*)

Sets when to realize a loss.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*realized_at_-liquidation*	flag when to realize a loss. FALSE: loss will be realized when the default occurs; TRUE: loss will be realized when liquidation occurs for Mortage/ABS collateral type deals or user-specified months prior liquidation for Student Loan collateral type deals.
in	*months_prior_-liquidation*	For Mortgage/ABS collateral type deals this input should be zero. For Student Loan deals, this input should be between 0 to recover lag of the asset to indicate in which month during default to liquidation to realize a the loss.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Other error
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* ptid = NULL;
*       // deal has been opened
*
*       int ret = set_realized_losses_at_liquidation(ptid, true, 1, false)
      ;
*       if(ret != 0)
*       {
*           //error handling
*       }
*
```

Note

if not set, the default value of realized losses at liquidation would be TRUE. months_prior_liquidation only work for student loan deals when realized losses at liquidation is TRUE, and the value should not be greater than the recovery delay of deal.

Warning

For Mortgage/ABS deals, regardless of the setting losses at liquidation, losses will always be realized at liquidation when the calculation method is JAPANESE_PREPAY_DEFAULT_PPYDEF or PREPAY_DEFAU-LT_BEFORE_SCHED_PRIN_PPYDEF

22.3.3.217 long CHASAPI set_recover_at_maturity_call (void ∗ *tid*, BOOLYAN *is_enabled*, BOOLYAN *set_sup_remic*)

Sets assumption recover defaults at Maturity/Call of Mortgage/ABS, with the ability to apply to underlying deals if it is a reremic.

Since

3.0.0

Availability SFW, CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*is_enabled*	If TRUE this will enable recover defaults at Maturity/Call. Otherwise it will not.
in	*set_sup_remic*	If TRUE this will replace any specific underlying deal settings. Otherwise it will not.

Return values

0	No error
-1	Error: Deal not opened
-99	Error: other errors, for details call get_deal_error_msg()

Example:

```
*        void* tid = NULL;
*        CMO_STRUCT *pCmo = new CMO_STRUCT;
*        memset(pCmo, 0, sizeof(CMO_STRUCT));
*        strcpy(pCmo->dealid,"AL2010-A");
*
*       // open deal
*        open_deal_ex(tid, pCmo);
*
*        // set recover defaults at Maturity/Call to true
*       double MDR = .0025;
*        set_defaults_ex(tid, DEFAULT_CURVE_MDR, 0, &MDR, -1, false);
*       set_service_advances_ex(tid,
      SERVICER_ADVANCES_BOTH, true);
*       set_recovery_lag_ex(tid, 3, -1, false);
*       double Recovery = .55;
*       set_recoveries_ex(tid, 0, &Recovery, -1, true);
*
*       set_recover_at_maturity_call(tid, true, false);
*
*       run_deal_ex(tid, pCmo);
*       close_deal_ex(tid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.218 long CHASAPI set_recovery_from (void ∗ *tid,* short *type,* BOOLYAN *set_sup_remic*)

This method can be used to specify whether to use the recovery rate at time of default or the recovery rate at time of recovery. It is only relevant when running a non-constant recovery rate vector and a non-zero recovery lag.

Since

1.5.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	Recovery rate applying method: • RECOVERY_RATE_AT_RECOVERY: Recovery rate at the time of recovery in case of recovery lag • RECOVERY_RATE_AT_DEFAULT: Recovery rate at the time of default in case of recovery lag.
in	*set_sup_remic*	Settings are applied to underlying deals if TRUE. Otherwise, it will not.

Return values

0	No error
-1	Deal not open
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Note

The default behavior for SFW deals is to use the recovery rate at the time of recovery, to be consistent with the CHS engine. One exception is when calling set_moodys_credit_model_settings(), which always uses recovery rate at the time of default (this method will not override that setting).

Example:

```
*   void* tid = NULL;
*   CMO_STRUCT cmos;
*   memset(&cmos, 0, sizeof(cmos));
*   set_input_path("C:\\Deals");
*   strcpy(cmos.dealid, "AL2010-A");
*
*   // open_deal_ex() must be called before set_recovery_from().
*   assert(-1 == set_recovery_from(tid,
      RECOVERY_RATE_AT_RECOVERY, true));
*   assert(0 == open_deal_ex(tid, &cmos));
*
*   // recovery rate at recovery
*   assert(0 == set_recovery_from(tid,
      RECOVERY_RATE_AT_RECOVERY, true));
*   // run deal
*   assert(0 == run_deal_ex(tid, &cmos));
*
*   // recovery rate at default
*   assert(0 == set_recovery_from(tid, RECOVERY_RATE_AT_DEFAULT,
      true));
*   // run deal
*   assert(0 == run_deal_ex(tid, &cmos));
*
*   close_deal_ex(tid, &cmos);
*
```

22.3.3.219 int CHASAPI set_reinvestment_type (void ∗ *tid,* short *reinv_type*)

Sets the reinvestment type that is used for reinvestment.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reinv_type*	Type to indicate which reinvestment settings to use, refer to enum REINV_TY-PE.

Return values

0	No error
-1	Error - Deal not opened
-2	Error - Invalid reinvestment type
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*     void *pDeal = NULL;
*     // deal is open
*
*     set_reinvestment_type(pDeal, GLOBAL_REINV);
*
```

22.3.3.220 void CHASAPI set_resec_exceptions_handling (RESEC_EXCEPTIONS_HANDLING *handling*)

Sets the handling of resec exceptions.

New feature Subject to change

Since

2.0.0

Availability CDOnet

Precondition

None.

Parameters

in	*handling*	The type of handle resec exceptions, refer to enum RESEC_EXCEPTIONS_-HANDLING.

Return values

void	

Example:

```
*      set_resec_exceptions_handling(
      RESEC_EXCEPTIONS_HANDLING_TREAT_AS_NONRESEC);
*
```

22.3.3.221 void CHASAPI set_resec_underlying_level (int *level*)

This method sets the underlying level of resec deals.

Since

2.0.2

Availability CDOnet

Precondition

None.

Parameters

in	*level*	The number of underlying levels will open/run. • input -1 means will open/run all level underlying deals. • input 0 means just open/run top deal. • input x>0 means open/run x levels of underlying deals.

Return values

void	

Example:

```
*       set_resec_underlying_level(-1);
*
```

Note

The function need to be called before calling open_deal_ex().

22.3.3.222 int CHASAPI set_service_advances_rates (void ∗ *tid,* int *group_number,* short *is_vector,* double ∗ *pval*)

This method sets service advance rate, either a vector or a constant, with ability to apply on different scope(pool group level or deal level).

Since

2.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
The service advance type has been set to SERVICER_ADVANCES_INTEREST or SERVICER_ADVANCE-S_BOTH with set_service_advances_ex().
The base of service advance rate have been set with set_service_advances_rates_type().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*group_number*	The collateral group number, 0 for total (deal level).
in	*is_vector*	0 for constant percentage (pval), length for double array pval.
in	*pval*	constant or vector in decimal for advances from servicer.

Return values

0	Success.
-1	Deal not open.
-2	Invalid parameter.
-99	Other error, use get_deal_error_msg() to see details.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == enable_sfw_delinq_projection(pDeal, true));
*       assert(0 == set_service_advances_ex(pDeal,
      SERVICER_ADVANCES_BOTH, true));
*       assert(0 == set_service_advances_rates_type(pDeal,
      SERVICER_ADVANCES_BASE_DEFAULT));
*
*       int group_number = 1;   // or 0 for all pools
*       double rates[] = {0.5};
*       assert(0 == set_service_advances_rates(pDeal, group_number, sizeof(rates)/
      sizeof(rates[0]), rates));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

set_service_advances_ex() set_service_advances_rates_type()

22.3.3.223 int CHASAPI set_service_advances_rates_type (void * *tid*, short *type*)

This method sets base of service advance rate, either default balance or delinquent balance.

Since

2.2.0

Availability SFW

Precondition

open_deal_ex() has been called.
The service advance type has been set to SERVICER_ADVANCES_INTEREST or SERVICER_ADVANCE-S_BOTH with set_service_advances_ex().

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type*	Setting advance assumption on delinquent balance or default balance, should be one of SERVICER_ADVANCES_BASE.

Return values

0	Success.
-1	Deal not open.
-2	Invalid parameter.
-99	Other error, use get_deal_error_msg() to see details.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_service_advances_ex(pDeal,
      SERVICER_ADVANCES_BOTH, true));
*
*       assert(0 == set_service_advances_rates_type(pDeal,
      SERVICER_ADVANCES_BASE_DEFAULT));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

See Also

set_service_advances_ex() set_service_advances_rates()

Note

If not set, servicer advance projections are based on SERVICER_ADVANCES_BASE_DEFAULT.

22.3.3.224 int CHASAPI set_service_reimburse_advint (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, BOOLYAN *reimburse_advint*)

Sets whether reimburse service advanced P&I for SFW and CDOnet deals.

Since

1.6.0

Availability SFW, CDOnet

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*reimburse_-advint*	TRUE/FALSE value to be set for reimbursing advanced P&I.

Return values

0	SUCCESS
-1	Error - Deal not opened
-99	Error - Invalid dso identifier (tid) and other errors

Example:

```
*   void* ptid = NULL;
*   //deal has been opened
*
*   int ret = set_service_reimburse_advint(ptid, null, TRUE);
*   if(ret != 0)
*   {
*       //error handling
*   }
*
```

Note

if not set, the default reimburse advanced P&I flag is TRUE.

Warning

when the deal service advance is SERVICER_ADVANCES_NOTHING, set_service_reimburse_advint cannot set reimburse_advint=TRUE.

22.3.3.225 void CHASAPI set_sfw_dll_num (const int & *num*)

This method sets the max number of sfw dll copies in RAM.

Since

2.8.0

Availability SFW

Precondition

None.

Parameters

in	*num*	The max sfw dll number to set.

Return values

None	

Example:

```
*      set_sfw_dll_num(8);
*
```

Note

- This function should be called before open_deal_ex
- For 32-bit system, the max dll number is capped by 16, and for 64-bit system the max dll number is capped by 4096.

22.3.3.226 void CHASAPI set_sfw_unload_flag (bool *unload_dll*)

Set the keep dll space flag to determine release dll space in memory or not when closing deals for SFW.

Since

3.3.0

Availability SFW

Parameters

in	*unload_dll*	Means release dll space in memory or not when closing deals. True means realse, False means not.

Return values

None.	

Example:

```
*        set_sfw_unload_flag(true);
*
```

22.3.3.227 int CHASAPI set_simulation_engine (void * *tid,* short *simulation_type*)

This method sets the simulation engine that users want to run.

Since

2.1.0

Availability CDOnet, SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*simulation_type*	The simulation engine that user want to run should be enums of SIMULATIO-N_TYPE.

Return values

0	Success.
-1	Deal not open.
-2	Invalid simulation type.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ACE06NC1");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*
*       assert(0 == set_simulation_engine(pDeal,
      SIMULATION_MONTE_CARLO));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.228 int CHASAPI set_smooth_losses (void ∗ *tid,* BOOLYAN *status,* BOOLYAN *set_sup_remic*)

This method sets the smooth losses flag.

Since

2.7.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*status*	Enable smooth losses or not • true means enable smooth losses • false means disable smooth losses
in	*set_sup_remic*	Settings are applied to underlying deals if TRUE. Otherwise, it will not.

Return values

0	Success.
-1	Deal not opened.
-99	Other errors.

Example:

```
*       //Deal has been opened
*       int ret = set_smooth_losses(pDeal, true, true);
*
```

Note

This function does not support student loan deals

22.3.3.229 int CHASAPI set_spot_spread (void ∗ *tid,* const char ∗ *currency,* ESG_RATING_TYPE *rating_type,* ESG_RATING_TERM *term_type,* int *num_paths,* short *rate_size,* double ∗∗ *idx_val*)

Sets spot spread rate that will be used for the calculation of bank loan call period.

Since

3.2.0

Availability CDOnet

Precondition

set_metrics_input_ex(), set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*currency*	The ISO name of the currency of the requested market index.
in	*rating_type*	ESG rating type.
in	*term_type*	ESG rating Term.
in	*num_paths*	Number of paths in the user input.
in	*rate_size*	The vector length of each path in two-dimensional array idx_val.
in	*idx_val*	A pointer to a two-dimensional array of interest rates in decimal.

Return values

0	No error.
-2	Error - Invalid currency ISO name.
-4	Error - Invalid path number.
-5	Error - Invalid rate vector size of each path.
-6	Error - Invalid rate pointer.
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void* pDeal = NULL;
*       //whole loan has been set.
*
*       const int path_number = 100;
*       METRIC_INPUT_STRUCT_EX metric_input_ex;
*       memset(&metric_input_ex, 0, sizeof(METRIC_INPUT_STRUCT_EX));
*       metric_input_ex.shift_amt = 0.005;
*       metric_input_ex.num_paths = path_number;
*       metric_input_ex.oas_mode = ENABLE_ALL;
*       assert(0 == set_metrics_input_ex(pDeal, &metric_input_ex));
*
*       const int rate_size = 200;
*       double **pVal = new double*[path_number];
*       for (int i = 0; i < path_number; ++i)
*       {
*           pVal[i] = new double[rate_size];
*       }
*       //Fill the index rate values for each path
*
*       ESG_RATING_TYPE rating_type = ESG_RATING_B;
*       ESG_RATING_TERM term_type = ESG_TERM_3M;
*       assert(0 == set_spot_spread(pDeal, "USD", rating_type, term_type, path_number,
       rate_size, pVal));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
*       for (int i = 0; i < path_number; ++i)
*            delete pVal[i];
*       delete[] pVal;
*
```

22.3.3.230 int CHASAPI set_trigger_override_ex (void ∗ *tid*, const char ∗ *reremic_deal_id_or_null*, const char ∗ *sub_trigger_name*, SBYTE *override_type*, int *override_date*)

This method overrides the requested sub-trigger.

Since

2.0.1

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*reremic_deal_id-_or_null*	The reremic deal id or null if not reremic.
in	*sub_trigger_-name*	The case-sensitive name of the requested sub-trigger be overridden.
in	*override_type*	The type of sub-trigger is overridden or will be overridden. 0 for no override, 1 for always "no" and 2 for always "yes".
in	*override_date*	The date from when the sub trigger status will be overridden. the format is "YYYYMMDD". It is would not set when the "override_type" is 0.

Return values

0	Success.
-1	Deal not open.
-2	Trigger not found.
-3	Invalid params.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       //Deal has been run
*
*       int ret = set_trigger_override_ex(pDeal, NULL, "PRO-RATA-1", 1, 20070701);
*       if(ret < 0)
*       {
*           //Error handle
*       }
*
```

Note

The trigger would be overridden by after calling run_deal_ex().

22.3.3.231 int CHASAPI set_up_ESG_model_interest_rates (ESG_MODEL_INPUTS ∗ *esg_inputs,* ESG_CURRENCY_RATE_INPUTS *esg_currency_inputs[ ],* int *esg_currency_inputs_size*)

Setup ESG model simulation interest rates, for OAS calculation, it would depends on the ESG model simulation interest rates.

Since

3.0.0

Availability All

Precondition

set_input_path() has been called.

Parameters

in	*esg_inputs*	ESG model inputs param.
in	*esg_currency_-inputs*	An array user allocated of the ESG currencies rates inputs.
in	*esg_currency_-inputs_size*	Array size of esg_currency_inputs

Return values

0	success, ESG output rates generated success.
<	0 Error - error

Example:

```
*     ESG_MODEL_INPUTS esgParam;
*     memset(&esgParam, 0, sizeof(ESG_MODEL_INPUTS));
*     strcpy(esgParam.ESGbhmPath,"C:\\installer\\8.5.0\\Models");
*     strcpy(esgParam.ESGLicenseFilePath,"C:\\installer\\Moody's Analytics-Internal
       Master Licence - For Staff Use Only-030717-011017.lic");
*     strcpy(esgParam.ESGCalibrationFilePath,"C:\\installer\\lmmp_testcalib_a.bhc");
*     strcpy(esgParam.ESGRatesOutputPath,"C:\\temp");
*     esgParam.YYYYMMDD = 20170801;
*     esgParam.SimulationPaths = 10;
*     esgParam.Periods = 500;
*     ESG_CURRENCY_RATE_INPUTS esgRateInput[2];
*     memset(esgRateInput, 0, sizeof(ESG_CURRENCY_RATE_INPUTS)*2);
*     strcpy(esgRateInput[0].Currency, "GBP");
*     strcpy(esgRateInput[1].Currency, "USD");
*     set_up_ESG_model_interest_rates(&esgParam, esgRateInput, 2);
*
```

Note

The order of the currencies array inputs may affect the ESG simulation output rates. The first currency in esg_currency_inputs would be the base economy of the ESG simulation.

Warning

ESG generates even number of paths for improved accuracy of the interest rate simulation. The last interest rate path from ESG would be ignored in OAS analysis if user sets ESG_MODEL_INPUTS::SimulationPaths to an odd number.

22.3.3.232 int CHASAPI set_whole_loan (void * *tid*, const WHOLE_LOAN_STRUCT * *whole_loan*, int *length*, int *initial_date*)

Set whole loan information

New feature Subject to change

Since

2.7.0

Availability SFW, CDOnet

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*whole_loan*	Whole loan detail information
in	*length*	Number of loans.
in	*initial_date*	Initial date of cash flow projection.

Return values

0	No error
-2	Error - Invalid factor for whole loan, use get_deal_error_msg() to see details.
-3	Error - Invalid length
-99	Error - Invalid dso identifier (tid) or other errors, for details call get_deal_error_-msg()

Example:

```
*       void *pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(pDeal, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(pDeal, CALC_LEVEL_FULL_WITH_LOAN, 1)
      ;
*
*
*      // get loan cashflow
*
```

22.3.3.233 int CHASAPI set_whole_loan_cumulative_rate (void ∗ *tid*, double *val*, long *loan_num*)

Set the cumulative default rate for loans/mortgages in whole loan analyzer. Under DEFAULT_PATTERN_BINARY mode, the cumulative default rate is at the portfolio level, representing the total portion of the starting PORTFOLIO balance that will default throughout the whole projection; Under DEFAULT_PATTERN_NONBINARY mode, it is at the per-loan level, representing the total portion of the starting LOAN balance that will default throughout the whole projection. The cumulative default rate setting only works at DEFAULT_PATTERN_BINARY and DEFAULT_PATTERN_NONBINARY mode, which can be set by set_whole_loan_default_method().

Since

3.1.0

Availability SFW

Precondition

set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*val*	The cumulative rate.
in	*loan_num*	The 1-based index of the loan or -1 to apply to all collateral in the deal.

Return values

0	Success.
-1	Error - set_whole_loan() not called.
-2	Error - Invalid loan number.
-3	Error - Other error.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(ptid, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(ptid, CALC_LEVEL_FULL_WITH_LOAN, 1);

*
*       // set whole loan cumulative rate.
*       int ret = set_whole_loan_cumulative_rate(ptid, 0.25, -1);
*
*       run_deal_ex(ptid, pCmo);
*       close_deal_ex(ptid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.234 int CHASAPI set_whole_loan_default_method (void * *tid*, WHOLE_LOAN_DEFAULT_METHOD_TYPE *type_index*)

Set the default amount calculation method for whole loan analyzer.Except mode "NORMAL", other modes also need "cumulative default rate" and "cumulative default pattern" being set as well.

Since

3.1.0

Availability SFW

Precondition

set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*type_index*	The specified method type, should be enums of WHOLE_LOAN_DEFAULT_-METHOD_TYPE.

Return values

0	Success.
-1	Error - set_whole_loan() not called.
-2	Error - Other error.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
```

```
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(ptid, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(ptid, CALC_LEVEL_FULL_WITH_LOAN, 1);

*
*       // set whole loan default method.
*       WHOLE_LOAN_DEFAULT_METHOD_TYPE type_index= DEFAULT_PATTERN_NONBINARY;
*       int ret = set_whole_loan_default_method(ptid, type_index);
*
*       run_deal_ex(ptid, pCmo);
*       close_deal_ex(ptid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.235 int CHASAPI set_whole_loan_default_timing (void ∗ *tid*, short *vector_length*, double ∗ *pval*)

Set the cumulative default pattern for whole loan analyzer.

Since

3.1.0

Availability SFW

Precondition

set_whole_loan() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*vector_length*	The length of the vector pointed to by pval.
in	*pval*	A pointer to the default pattern(timing). Value for current period (0-indexed element) will not be applied.

Return values

0	Success.
-1	Error - set_whole_loan() not called.
-2	Error - Other error.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* ptid= NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*
*       std::vector<WHOLE_LOAN_STRUCT> loans;
*       // set informations for each loan in vector loans
*       set_whole_loan(ptid, &loans.front(), 10, 20160101);
*
*       set_deal_calc_level(ptid, CALC_LEVEL_FULL_WITH_LOAN, 1);

*
*       // set whole loan default timing.
*       double value = 0.1;
*       int ret = set_whole_loan_default_timing(ptid, 1, &value);
*
*       run_deal_ex(ptid, pCmo);
*       close_deal_ex(ptid, pCmo);
*       delete pCmo;
*       pCmo = NULL;
*
```

Note

Vector_length input longer than the actual size of pval would cause projection issue. The first element value would not be used in pval in projection; so recommended to set to "0".

22.3.3.236 int CHASAPI use_spot_values_for_initial_period (void * *tid*, bool *enable*)

This method is to use spot values for initial period to be consistent with PA credit model for SFW student loan deals.

Since

3.6

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	No error.
-1	Error - Deal not open.
-99	Error - Call get_deal_error_msg() for detail.

Example:

```
*       void* pDeal = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "ABF00001");
*
*       set_engine_preference(
      PICK_SFW_ENGINE_FOR_MAPPED_DEALS);
*       assert(0 == open_deal_ex(pDeal, pCmo));
*       assert(0 == use_spot_values_for_initial_period(pDeal, true));
*
*       assert(0 == close_deal_ex(pDeal, pCmo));
*       delete pCmo;
*       pCmo = NULL;
*
```

22.3.3.237 long CHASAPI view_moodys_student_loan_info (void * *tid*, short *index*, MOODYS_STUDENT_LOAN_INFO *all_colls[ ]*, short *length*)

This method retrieves the additional descriptive student loan information of a piece of collateral specified by index.

Since

1.6.0

Availability SFW

Precondition

open_deal_ex() has been called.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*index*	The 0-based index of a piece of collateral (-1 for all collateral).
in	*length*	The size of the MOODYS_STUDENT_LOAN_INFO structure.
out	*all_colls[]*	A client-allocated array of MOODYS_STUDENT_LOAN_INFO structures.

Return values

>=0	No error.
-1	Error - Deal not opened.
-3	Error - Invalid loan index.
-4	Error - Invalid size.
-5	Error - No output vector passed.
-99	Error - Other error, call get_deal_error_msg() for detail.

Example:

```
*      void* ptid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT;
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "WSLT2006-1");
*
*      open_deal_ex(ptid, pCmo);
*
*      MOODYS_STUDENT_LOAN_INFO coll[1] = {0};
*      int ret = view_moodys_student_loan_info(ptid, 1, coll, sizeof(
      MOODYS_STUDENT_LOAN_INFO));
*      if(ret < 0)
*      {
*          // error handling
*      }
*
*      MOODYS_STUDENT_LOAN_INFO *all_coll = new
      MOODYS_STUDENT_LOAN_INFO[pCmo->num_colls];
*      ret = view_moodys_student_loan_info(ptid, -1, all_coll, sizeof(
      MOODYS_STUDENT_LOAN_INFO));
*      if(ret < 0)
*      {
*          // error handling
*      }
*
*      delete [] all_coll;
*      all_coll = NULL;
*      close_deal_ex(ptid, pCmo);
*      delete pCmo;
*      pCmo = NULL;
*
```

Note

If all collateral is requested the arrays all_colls[] must be allocated to be at least as long as the value CMO_-STRUCT.num_colls returned by open_deal().

22.4 include/WSAAdcoProviderApi.h File Reference

Data Structures

- struct MarkitAdcoPrepayModelDials

  The fine tune parameters for the ADCO prepay model. This supplements structure MarkitAdcoTuningParam, and is optional.
- struct MarkitAdcoDefaultModelDials

  The fine tune parameters for the ADCO default model. This supplements structure MarkitAdcoTuningParam, and is optional.
- struct MarkitAdcoTuningParam

  The main tuning parameters for the ADCO default model.

- struct MarkitAdcoScenarioParams

  The Scenario parameters users can change from run to run, see ResetADCOScenario().

Functions

- int ADCO_PROVIDER_API SetupADCOModel (void ∗tid, MarkitAdcoTuningParam ∗tuning)
- int ADCO_PROVIDER_API RemoveADCOModel (void ∗tid)
- int ADCO_PROVIDER_API ResetADCOScenario (void ∗tid, MarkitAdcoScenarioParams ∗scenParam)
- int ADCO_PROVIDER_API ResetADCOInterestRates (void ∗tid)
- int ADCO_PROVIDER_API ResetADCOHpiRates (void ∗tid, int HPI_vector_size, double ∗HPI_vector)
- int ADCO_PROVIDER_API GetADCOVersion (void ∗tid, char ∗version, int size)
- int ADCO_PROVIDER_API SetTuningParam (void ∗tid, const char ∗paramName, double paramValue, char ∗error_message, int max_size_of_error_message)
- int ADCO_PROVIDER_API GetCurrentLoanType (void ∗tid, char ∗loanType, int max_size_of_loan_type, char ∗error_message, int max_size_of_error_message)

22.4.1 Function Documentation

22.4.1.1 int ADCO_PROVIDER_API GetADCOVersion (void ∗ *tid,* char ∗ *version,* int *size*)

Get The ADCO version.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
out	*version*	The buffer to store ADCO version.
in	*size*	The buffer size of the output pointer 'version'.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       char version[10];
*       int size = 10;
*       GetADCOVersion(tid, version, size);
*
```

22.4.1.2 int ADCO_PROVIDER_API GetCurrentLoanType (void ∗ *tid,* char ∗ *loanType,* int *max_size_of_loan_type,* char ∗ *error_message,* int *max_size_of_error_message*)

Get the type of loan that are dealed with. This function should be called from LOAN_TUNNING_CB.Used internally for debugging; Soon to be deprecated.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*max_size_of_-loan_type*	The max size of user allocated array loanType.
in	*max_size_of_-error_message*	The max size of user allocated array error_message.
out	*loanType*	A pointer to client-allocated char array.
out	*error_message*	A pointer to client-allocated char array.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       char loanType[100];
*       int max_size_of_loan_type = 100;
*       char error_message[200];
*       int max_size_of_error_message = 200;
*       GetCurrentLoanType(tid, loanType, max_size_of_loan_type, error_message,
      max_size_of_error_message);
*
```

22.4.1.3 int ADCO_PROVIDER_API RemoveADCOModel (void ∗ *tid*)

Uninstall ADCo model from WSA API.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAdcoTuningParam AdcoParam;
*       MarkitAdcoPrepayModelDials pDials;
*       AdcoParam.prepayModelDials = &pDials;
*       SetupADCOModel(tid,&AdcoParam);
*       run_deal_ex(tid, pCmo);
*
*       RemoveADCOModel(tid);
*
```

22.4.1.4 int ADCO_PROVIDER_API ResetADCOHpiRates (void ∗ *tid,* int *HPI_vector_size,* double ∗ *HPI_vector*)

Reset HPI rates in ADCo model.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*HPI_vector_size*	The length of user allocated array HPI_vector.
in	*HPI_vector*	A pointer to client-allocated HPI_vector array.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAdcoTuningParam AdcoParam;
*       MarkitAdcoPrepayModelDials pDials;
*       AdcoParam.prepayModelDials = &pDials;
*       SetupADCOModel(tid,&AdcoParam);
*
*       int HPI_vector_size = 10;
*       double HPI_vector[10];
*       ResetADCOHpiRates(tid, HPI_vector_size, HPI_vector);
*
```

22.4.1.5 int ADCO_PROVIDER_API ResetADCOInterestRates (void ∗ *tid*)

Reset all interest rates in ADCo model.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAdcoTuningParam AdcoParam;
*       MarkitAdcoPrepayModelDials pDials;
*       AdcoParam.prepayModelDials = &pDials;
*       SetupADCOModel(tid,&AdcoParam);
*
*       ResetADCOInterestRates(tid);
*
```

22.4.1.6 int ADCO_PROVIDER_API ResetADCOScenario (void ∗ *tid,* MarkitAdcoScenarioParams ∗ *scenParam*)

Reset all interest rates and set the parameters in ADCo model.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*scenParam*	A pointer to a client-allocated MarkitAdcoScenarioParams structure.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAdcoTuningParam AdcoParam;
```

```
 *      MarkitAdcoPrepayModelDials pDials;
 *      AdcoParam.prepayModelDials = &pDials;
 *      SetupADCOModel(tid,&AdcoParam);
 *
 *      MarkitAdcoScenarioParams params;
 *      ResetADCOScenario(tid,&params);
 *
```

22.4.1.7 int ADCO_PROVIDER_API SetTuningParam (void * *tid*, const char * *paramName*, double *paramValue*, char * *error_message*, int *max_size_of_error_message*)

Set tuning parameters. This function should be called from LOAN_TUNNING_CB.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*paramName*	The tuning parameter name in ADCo model.
in	*paramValue*	The tuning parameter value.
in	*max_size_of_-error_message*	The max size of user allocated array error_message.
out	*error_message*	A pointer to client-allocated array for storing error message.

Return values

0	Success.
other	Fail.

Example:

```
 *      void *tid = NULL;
 *      CMO_STRUCT *pCmo = new CMO_STRUCT();
 *      memset(pCmo, 0, sizeof(*pCmo));
 *      strcpy(pCmo->dealid, "SAS059XS");
 *      open_deal_ex(tid, pCmo);
 *
 *      const int ERR_SIZE = 200;
 *      char error[ERR_SIZE] = {0};
 *      SetTuningParam(tid, "SmmTuneAge", 5.0, error, ERR_SIZE);
 *
```

22.4.1.8 int ADCO_PROVIDER_API SetupADCOModel (void * *tid*, MarkitAdcoTuningParam * *tuning*)

Install ADCo model to WSA API.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*tuning*	A pointer to a client-allocated MarkitAdcoTuningParam structure.

Return values

0	Success.
other	Fail.

Example:

```
 *      void *tid = NULL;
 *      CMO_STRUCT *pCmo = new CMO_STRUCT();
 *      memset(pCmo, 0, sizeof(*pCmo));
 *      strcpy(pCmo->dealid, "SAS059XS");
 *      open_deal_ex(tid, pCmo);
 *
 *      MarkitAdcoTuningParam AdcoParam;
 *      MarkitAdcoPrepayModelDials pDials;
 *      AdcoParam.prepayModelDials = &pDials;
 *      SetupADCOModel(tid,&AdcoParam);
 *      run_deal_ex(tid, pCmo);
 *
```

22.5 include/WSAAftProviderApi.h File Reference

Data Structures

- struct MarkitAftDefaultModelDials

  The fine tune parameters for the AFT default model. This supplements structure MarkitAftTuningParam, and is optional.
- struct MarkitAftPrepayModelDials

  The fine tune parameters for the AFT prepay model. This supplements structure MarkitAftTuningParam, and is optional.
- struct MarkitAftTuningParam

  The main tuning parameters for the AFT default model.
- struct MarkitAftScenarioParams

  The Scenario parameters users can change from run to run, see ResetAFTScenario().

Functions

- int AFT_PROVIDER_API SetupAFTModel (void ∗tid, MarkitAftTuningParam ∗tuning)
- int AFT_PROVIDER_API RemoveAFTModel (void ∗tid)
- int AFT_PROVIDER_API ResetAFTScenario (void ∗tid, MarkitAftScenarioParams ∗scenParam)
- int AFT_PROVIDER_API ResetAFTInterestRates (void ∗tid)
- int AFT_PROVIDER_API ResetAFTHpiRates (void ∗tid, int HPI_vector_size, double ∗HPI_vector)
- int AFT_PROVIDER_API SetPrepayModelDials (void ∗tid, MarkitAftPrepayModelDials ∗prepayModelDials)
- int AFT_PROVIDER_API SetDefaultModelDials (void ∗tid, MarkitAftDefaultModelDials ∗defaultModelDials)

22.5.1 Function Documentation

22.5.1.1 int AFT_PROVIDER_API RemoveAFTModel (void ∗ *tid*)

Uninstall AFT model from WSA API.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	Success.
other	Fail.

Example:

```
*      void *tid = NULL;
*      CMO_STRUCT *pCmo = new CMO_STRUCT();
*      memset(pCmo, 0, sizeof(*pCmo));
*      strcpy(pCmo->dealid, "SAS059XS");
*      open_deal_ex(tid, pCmo);
*
*      MarkitAftTuningParam AftParam;
*      MarkitAftPrepayModelDials pDials;
*      AftParam.prepayModelDials = &pDials;
*      SetupAFTModel(tid,&AftParam);
*      run_deal_ex(tid, pCmo);
*
*      RemoveAFTModel(tid);
*
```

22.5.1.2 int AFT_PROVIDER_API ResetAFTHpiRates (void ∗ *tid,* int *HPI_vector_size,* double ∗ *HPI_vector*)

Reset HPI rates in AFT model.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*HPI_vector_size*	The length of user allocated array HPI_vector.
in	*HPI_vector*	A pointer to client-allocated HPI_vector array.

Return values

0	Success.
other	Fail.

Example:

```
*     void *tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT();
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "SAS059XS");
*     open_deal_ex(tid, pCmo);
*
*     MarkitAftTuningParam AftParam;
*     MarkitAftPrepayModelDials pDials;
*     AftParam.prepayModelDials = &pDials;
*     SetupAFTModel(tid,&AftParam);
*
*     int HPI_vector_size = 10;
*     double HPI_vector[10];
*     ResetAFTHpiRates(tid, HPI_vector_size, HPI_vector);
*
```

22.5.1.3 int AFT_PROVIDER_API ResetAFTInterestRates (void ∗ *tid*)

Reset all interest rates in AFT model.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.

Return values

0	Success.
other	Fail.

Example:

```
*     void *tid = NULL;
*     CMO_STRUCT *pCmo = new CMO_STRUCT();
*     memset(pCmo, 0, sizeof(*pCmo));
*     strcpy(pCmo->dealid, "SAS059XS");
*     open_deal_ex(tid, pCmo);
*
*     MarkitAftTuningParam AftParam;
*     MarkitAftPrepayModelDials pDials;
*     AftParam.prepayModelDials = &pDials;
*     SetupAFTModel(tid,&AftParam);
*
*     ResetAFTInterestRates(tid);
*
```

22.5.1.4 int AFT_PROVIDER_API ResetAFTScenario (void ∗ *tid*, MarkitAftScenarioParams ∗ *scenParam*)

Reset all interest rates and set the parameters in AFT model.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*scenParam*	A pointer to a client-allocated MarkitAftScenarioParams structure.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAftTuningParam AftParam;
*       MarkitAftPrepayModelDials pDials;
*       AftParam.prepayModelDials = &pDials;
*       SetupAFTModel(tid,&AftParam);
*
*       MarkitAftScenarioParams params;
*       ResetAFTScenario(tid,&params);
*
```

22.5.1.5 int AFT_PROVIDER_API SetDefaultModelDials (void ∗ *tid,* MarkitAftDefaultModelDials ∗ *defaultModelDials*)

Set default model dials in AFT.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*defaultModel-Dials*	The pointer of the MarkitAftDefaultModelDials object.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAftDefaultModelDials   defaultModelDials;
*       SetDefaultModelDials(tid, &defaultModelDials);
*
```

22.5.1.6 int AFT_PROVIDER_API SetPrepayModelDials (void ∗ *tid,* MarkitAftPrepayModelDials ∗ *prepayModelDials*)

Set prepay model dials in AFT.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*prepayModel-Dials*	The pointer of the MarkitAftPrepayModelDials object.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAftPrepayModelDials aftPrepDials;
*       SetPrepayModelDials(tid,&aftPrepDials);
*
```

22.5.1.7 int AFT_PROVIDER_API SetupAFTModel (void ∗ *tid*, MarkitAftTuningParam ∗ *tuning*)

Install AFT model to WSA API.

Parameters

in	*tid*	The deal/scenario object identifier. Null if using non-thread-safe calls.
in	*tuning*	A pointer to a client-allocated MarkitAftTuningParam structure.

Return values

0	Success.
other	Fail.

Example:

```
*       void *tid = NULL;
*       CMO_STRUCT *pCmo = new CMO_STRUCT();
*       memset(pCmo, 0, sizeof(*pCmo));
*       strcpy(pCmo->dealid, "SAS059XS");
*       open_deal_ex(tid, pCmo);
*
*       MarkitAftTuningParam aftParam;
*       MarkitAftPrepayModelDials pDials;
*       aftParam.prepayModelDials = &pDials;
*       SetupAFTModel(tid, &aftParam);
*       run_deal_ex(tid, pCmo);
*
*
```

Index