THE RISK-BASED CAPITAL SIMULATION APPLICATION TECHNICAL REFERENCE MANUAL

OFFICE OF FEDERAL HOUSING ENTERPRISE OVERSIGHT
1700 G STREET, NW
4TH FLOOR
WASHINGTON, DC 20552

31 October, 2008

CONTENTS

Figures

Tables

1. INTRODUCTION

1.1 About This Manual

1.1.1 Audience

This manual is intended for the Software Engineer with software development experience on the UNIX operating system.

1.1.2 Manual Overview

This manual lists all of the RBCSIM application components required to compile and build the RBCSIM executables. This manual also includes high-level object-model diagrams. --- Change by Lessans ---

1.1.3 Additional Documentation

In addition to this manual, the RBCSIM application includes the following printed and on-line documentation:

- The Risk-Based Capital Simulation Application Installation Manual.
- The Risk-Based Capital Simulation Application User Manual.
- The Risk-Based Capital Report Instructions.
- The Risk-Based Capital Stylized Data Set Overview.
- OFHEO Phase 2 Business Rules.
- CreditEnhancementContract-Phase3.
- MultifamilyBusinessRules-Phase3.
- NMI Business Rules-Phase3.
- SingleClassMBSBusRules-Phase3.
- SingleFamilyBusinessRules-Phase3.
- Additional documentation is available at the official OFHEO web site: http://www.ofheo.gov.

1.1.4 Contact Point

All questions regarding the RBCSIM application should be emailed to: rbcquestions@ofheo.gov .

1.2 About OFHEO

The Housing and Community Development Act of 1992, under Title XIII, the Federal Housing Enterprises Financial Safety and Soundness Act of 1992, established the Office of Federal Housing Enterprise Oversight (OFHEO). The primary function of the Office is to perform financial regulation of Fannie Mae and Freddie Mac (collectively referred to as the "Enterprises") to ensure that the Enterprises are adequately capitalized and operating safely, in accordance with the Act.

1.2.1 OFHEO's Mission

OFHEO was required by the Federal Housing Enterprises Financial Safety and Soundness Act of 1992 (1992 Act) to establish minimum and risk-based capital standards as part of its role as safety and soundness regulator. These capital requirements are intended to ensure both Enterprises continue to operate and perform their crucial roles in the secondary mortgage market, keeping constant the flow of funds to mortgage lenders and prospective American homeowners. By ensuring the Enterprises are adequately capitalized, OFHEO minimizes the risk that American taxpayers will ever be asked to pay for losses at these complex financial institutions.

OFHEO evaluates capital adequacy from other perspectives as well. OFHEO's **examination program** conducts continuous, comprehensive examinations of the Enterprises to ensure they are operating under standards of financial safety and soundness. OFHEO's examination of the Enterprises provides a qualitative assessment of capital adequacy. The more direct and quantitative tools are OFHEO's **minimum** and **risk-based capital** standards, which are supplemented with other tests and analyses. OFHEO's minimum capital standard is calculated based on specific percentages for assets and off-balance sheet guarantees. The minimum capital level is therefore determined more by the size of the Enterprise than its specific risks. The risk-based standard, in contrast, requires that Fannie Mae and Freddie Mac each have enough capital to survive prolonged, severe problems in financial and economic markets, as well as management or operational failures. It is directly related to the risks the Enterprises are exposed to in their current business.

1.2.2 The Risk-Based Capital Rule

The risk-based capital regulation meets the specific requirements of the 1992 Act. The rule utilizes a stress test to determine the amount of capital needed to protect against credit and interest rate risks, and requires 30 percent additional capital to protect against unspecified management and operations risk. The regulation itself is the blueprint needed to construct the stress test and calculate the risk-based capital requirement for Freddie Mac and Fannie Mae. It is a detailed description of the stress test allowing the Enterprises and others to essentially replicate the stress test, as required by law.

What is the Stress Test?
OFHEO's risk-based capital standard is based on a 10-year stress test. A stress test measures risk in the context of a company's overall portfolio, including the effectiveness of a company's risk management strategies. While companies often use stress tests for internal risk management, and rating agencies use stress tests to rate companies and securities, OFHEO is among the first financial institution regulators to use its own stress test to determine capital

adequacy.

OFHEO's stress test simulates an Enterprise's financial performance over a 10-year period under severe economic conditions. Key aspects of the severe economic conditions used in OFHEO's stress test are defined in the 1992 Act and further specified in OFHEO's risk-based capital regulation. These conditions include high levels of mortgage defaults, with associated losses and large, sustained movements in interest rates, both increasing (up-rate scenario) and decreasing (down-rate scenario).

OFHEO uses a detailed computer model to simulate each Enterprise's cash flows associated with mortgages and other financial assets and obligations under the severe economic conditions of the stress test. The modeling of incoming and outgoing cash flows captures the risks embedded in those financial assets and obligations and the benefits of the hedges each Enterprise has set in place. To meet OFHEO's risk-based capital standard, each Enterprise must have sufficient capital to support any losses generated under these severe economic conditions plus an additional 30 percent for unspecified management and operations risks. The result is a stringent test of the capital adequacy of each Enterprise.

1.3 Software Identification

RBCSIM 10/31/2008 Release.

2. SOFTWARE SUMMARY

2.1 Software Application Summary

The RBCSIM Software is composed of two subsystems: the data validation system and the simulation model. Both subsystems use a Sybase database for storing input data and final results.

2.1.1 Data Validation System

The Data Validation System (DVS) performs a variety of checks to ensure the input data is model ready. The DVS checks for referential integrity, compliance with allowable values and the internal consistency of the fields comprising each record using a variety of 'business rules'. As the simulation model performs a limited set of data quality checks, it is very important that new data sets are validated prior to running them through the simulation model. Running the simulation model with data that has not been validated may result in inaccurate results or application instability.

The DVS is implemented using a combination of Perl and SAS scripts and Sybase stored procedures. Information on installing and configuring the DVS can be found in section 4.2 of *Risk-Based Capital Simulation Application Installation Manual*. Information on running the DVS can be found in section 3.1 of the *Risk-Based Capital Simulation Application User Manual*.

2.1.2 Simulation Model

The simulation model calculates the RBC Capital Requirement. The model is composed of three modules that project cash flows and a module that performs accounting, tax, investment, funding, financial reporting and capital calculations. All of the modules are implemented in C++ and are executed from the command line using a set of configuration files. These configuration files contain both application configuration information as well as policy parameters. More information on the individual modules follows:

Whole Loan Cash Flow. The Whole Loan Cash Flow (WLCF) module projects cash flows for retained and sold whole loans as well as for commitments. The WLCF reads information from the configuration file and input data from the database, performs amortization, default and prepayment calculations and writes its output to a text file as projected cash flows. Sections 3.2 and 3.6 of the RBC Rule provide a detailed discussion of the calculations performed by the WLCF module. The WLCF operates in two modes. One mode produces cash flows for retained and sold whole loans (WLCF); the other mode produces cash flows for commitments (CMT).

Mortgage Related Securities. The Mortgage Related Securities (MRS) module projects cash flows for single class MBS, Mortgage Revenue Bonds (MRBs) and multi-class MBS (e.g. REMICs, Strips, etc.). The MRS module uses the proprietary Intex Solutions, Inc. API to project the cash flows for multi-class MBS. The MRS reads information from the configuration file and input data from the database, performs amortization, default and prepayment calculations and writes its output to a text file as projected cash flows. Section 3.7 of the RBC Rule provides a detailed discussion of the calculations performed by the MRS module. The MRS operates in three modes. One mode produces cash flows for single class MBS (MBS), the second mode produces cash flows for MRBs (MRB) and the third mode produces cash flows for multi-class MBS (REMIC).

Non-Mortgage Instruments. The Non-Mortgage Instruments (NMI) module projects cash flows for debt instruments, non-mortgage related investments, guaranteed investment contracts, preferred stock, and derivative contracts. The NMI module also performs the Alternative Modeling Treatment (AMT) calculations. The NMI module uses the proprietary Intex Solutions, Inc. API to project the cash flows for non-mortgage-related ABS (e.g. auto loans, credit cards, etc.). The NMI module reads information from the configuration file and input data from the database, performs financial calculations and writes its output to a text file as projected cash flows. Sections 3.8 and 3.9 of the RBC Rule provide a detailed discussion of the calculations performed by the NMI module. The NMI operates in three modes. One mode produces cash flows for Futures (FUT), the second mode produces cash flows for items subject to AMT (AMT), and the third mode produces cash flows for all other instruments (NMI).

Reporting and Decisions Module. The Reporting and Decisions Module (RDM) implements the accounting, tax, investment, funding, financial reporting and capital calculation processes. The RDM module reads information from the configuration file, input data from the database, and the output files from the three cash flow modules, performs financial and accounting calculations and writes output to the database. Sections 3.10 and 3.12 of the RBC Rule provide a detailed discussion of the calculations performed by the RDM module.

Information on installing and configuring the simulation module can be found in section 4.2 of *Risk-Based Capital Simulation Application Installation Manual*. Information on running the simulation model can be found in section 3.2 of the *Risk-Based Capital Simulation Application User Manual.*

2.2 RBC Application Workflow

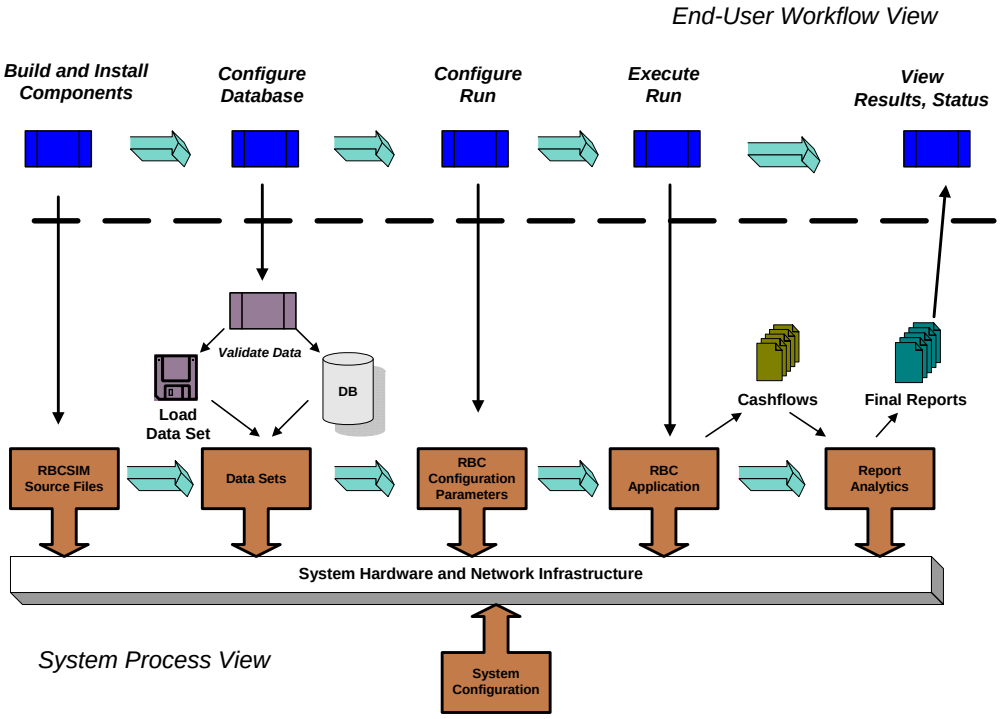
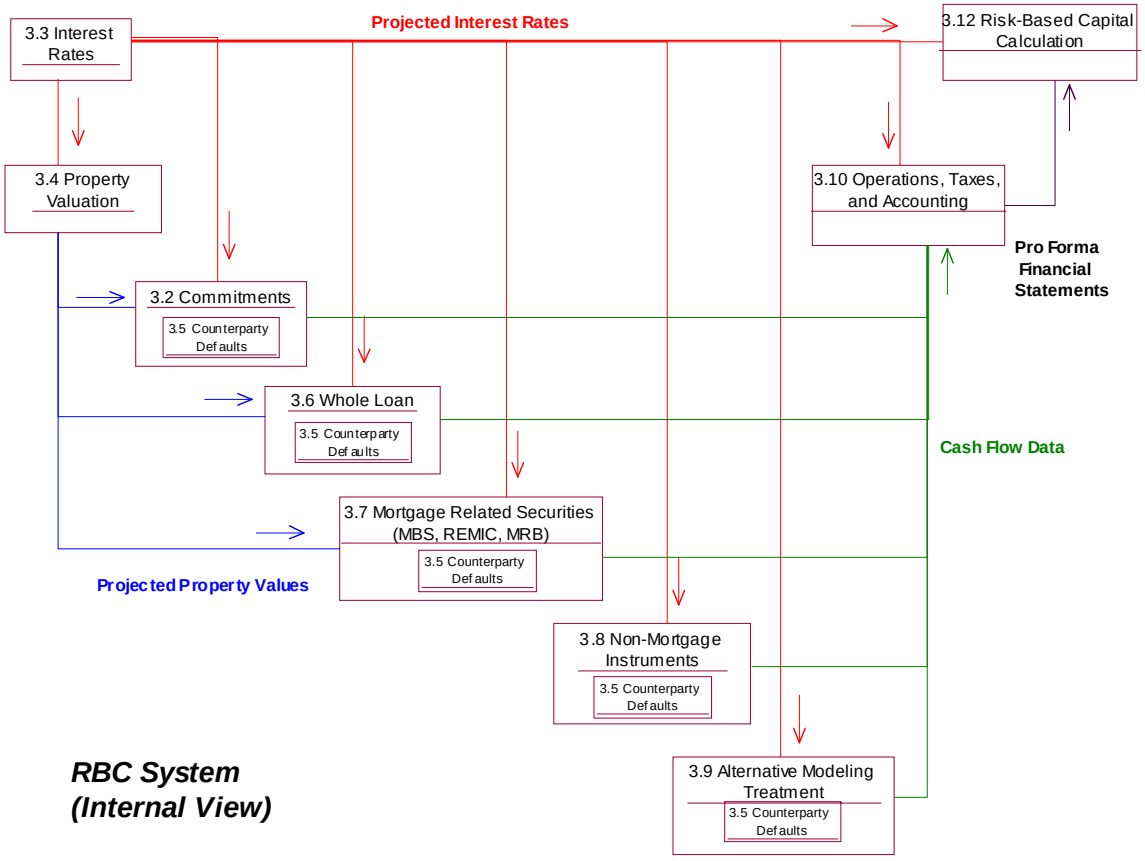


Figure 2-1: RBCSIM Workflow

2.3 RBC Application Rule Information Flow



2.4 Software Component List

This section identifies the software files, including database and data files that must be installed for the software to operate.

2.4.1 RBCSIM Database Creation and Output Scripts

Table 2-1: RBCSIM Database Create, Set-up and Load Scripts

Script Name	Path Location	Purpose
RBC_RDM_DDL.sql	$RBC_HOME/db/database/bin	To create the database objects associated with the Report Decision Module (RDM) and the Risk-based Capital (RBC) module.
RBC_RDM_grants.sql	$RBC_HOME/db/database/bin	To grant access to the database objects.
RBC_RDM_load_Acme_data.bcp	$RBC_HOME/db/database/bin	To load the data tables with Acme (stylized) data using the bulk-copy (bcp) utility.
RBC_RDM_load_Acme_lookup.bcp	$RBC_HOME/db/database/bin	To load the data tables with look-up data using the bulk-copy (bcp) utility.

Table 2-2: RBCSIM Database Create, Set-up and Load Output Scripts

Script Name	Path Location	Purpose
RBC_RDM_DDL_example.txt	$RBC_HOME/db/database/output	To demonstrate a successful execution of the *RBC_RDM_DDL.sql* script. Note: the scripts may not be identical due to technical environmental differences and data segment specifications.
RBC_RDM_grants_example.txt	$RBC_HOME/db/database/output	To demonstrate a successful execution of the *RBC_RDM_grants.sql* script. Note: the scripts may not be identical due to technical environmental differences.
RBC_RDM_bcp_Acme_data_example.txt	$RBC_HOME/db/database/output	To demonstrate a successful execution of the *RBC_RDM_load_Acme_data.bcp* script based on the stylized data provided. Note: the scripts may not be identical due to technical environmental differences including the amount of time it takes to load the data.
RBC_RDM_bcp_Acme_lookup_example.txt	$RBC_HOME/db/database/output	To demonstrate a successful execution of the *RBC_RDM_load_Acme_lookup.bcp* script based on the data provided. Note: the scripts may not be identical due to technical environmental differences including the amount of time it takes to load the data.

2.4.2 Data Validation Database, System and Output Scripts

Table 2-3: Data Validation Database Create, Set-up and Load Scripts

Script Name	Path Location	Purpose
Data_Validation_System_v1.0_ddl.sql	$RBC_HOME/db/data_validation_system/bin	To create the database objects associated with Data Validation System (DVS) module.
DVS_load_data.bcp	$RBC_HOME/db/data_validation_system/bin	To load the lookup table with data using the bulk-copy (bcp) utility.
DVS_business_rules.pro	$RBC_HOME/db/data_validation_system/bin	To create the Sybase stored procedures used to perform primary key, referential integrity and business rule data checks.
DVS_grants.sql	$RBC_HOME/db/data_validation_system/bin	To grant access to the database objects.
RBC_Report_Instr_Format_Lib.sas	$RBC_HOME/db/data_validation_system/bin	To create the SAS Format Library used to support column level allowable value checks.
validation_rule_lookup.dat (data file)	$RBC_HOME/db/data_validation_system/data	Data file used to populate the *validation_rule_lookup* table.

Table 2-4: Data Validation System Scripts

Script Name	Path Location	Purpose
scan_data.pl	$RBC_HOME/db/data_validation_system/bin	To search all of the data files contained in the directory for control characters or unwanted spaces.
check_first_coupon_date.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs first coupon date validation.
check_iss_date.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs issue date validation.
check_mty_date.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs maturity date validation.
check_valid_date.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs ad-hoc date validation.
check_vals.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that compares the reported value to the allowable values contained in the SAS Format library.
create_date_format.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that creates the valid Report Date value for a given quarter.
final_report_insert.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that produces the final allowable value violation report (in SAS) and inserts the violation(s) into the DVS data tables.
main_ffo.sas	$RBC_HOME/db/data_validation_system/bin	Main (or "master") SAS Program that kicks off the column level allowable value validations invoking macros as needed. Run-time parameters need to be entered prior to running the program.
mamt_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the AMT table.
marm_cmt_data_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the ARM Related Data Elements – Commitments table.

Script Name	Path Location	Purpose
marm_related_data_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the ARM Related Data Elements table.
mce_cmt_data_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the CE Data Elements – Commitments table.
mce_contract_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the CE Contract Elements table.
mce_data_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the CE Data Elements table.
mcntrprty_cr_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Counterparty Credit Ratings table.
mcntrprty_lkp_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Counterparty Look-up table.
mfin_instmt_mstr_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Financial Instrument Master table.
Mfutures_options_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Futures/Options table.
midx_formula_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Index Formula table.
minstmt_cred_rating_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Instrument Credit Rating table.
minstnm_assn_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Instrument Association table.
mint_paymt_formula_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Instrument Payment Formula table.
mint_paymt_sched_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Instrument Payment Schedule table.
mmbs_arm_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the MBS ARM table.
mmbs_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the MBS table.
mmf_data_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the MF Data Elements table.
mmrb_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the MRB table.
mmulti_class_derv_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the MultiClass Derivatives table.
moption_sched_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Option Schedule table.
mota_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the OTA table.
mperf_hist_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Performance History table.
mprin_chng_sched_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Principle Change Schedule table.
mrbc_reconciliation_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the RBC Reconciliation table.
mreference_asset_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Reference Asset table.

Script Name	Path Location	Purpose
msf_cmt_data_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the SF Data Elements – Commitments table.
msf_data_elements_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the SF Data Elements table.
mtrade_hist_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Trade History table.
mwl_cmt_master_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Whole Loan Master - Commitments table.
mwl_master_ffo.sas	$RBC_HOME/db/data_validation_system/bin	SAS macro that performs column level validation on the Whole Loan Master table.
DVS_run_data_checks.sql	$RBC_HOME/db/data_validation_system/bin	To execute primary key, referential integrity and business rule data checks. Run-time parameters need to be entered prior to running the program.

Table 2-5: Data Validation Output Scripts

Script Name	Path Location	Purpose
DVS_v1.0_ddl_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *Data_ Validation_System_v1.0_ddl.sql* script. Note: the scripts may not be identical due to technical environmental differences and data segment specifications.
DVS_bcp_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *DVS_load_data.bcp* script. Note: the scripts may not be identical due to technical environmental differences including the amount of time it takes to load the data.
DVS_business_rules_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *DVS_business_rules.pro* script. Note: the scripts may not be identical due to technical environmental differences and data segment specifications.
DVS_grants_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *DVS_grants.sql* script. Note: the scripts may not be identical due to technical environmental differences.
RBC_format_lib_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *RBC_Report_Instr_Format_Lib.sas* script. Note: the scripts may not be identical due to technical environmental differences, the version of SAS used and the storage location of the SAS Format Library.
log_fil_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *scan_data.pl* script. Note: the scripts may not be identical due to technical environmental differences.
main_ffo_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *main_ffo.sas* script based on the stylized data provided. Note: the scripts may not be identical due to technical environmental differences, the version of SAS used, the location of the SAS Format Library and parameter specifications.
DVS_run_data_checks_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a successful execution of the *DVS_run_data_checks.sql* script. Note: the scripts may not be identical due to technical environmental differences

		and parameter specifications.

Table 2-6: Data Validation Report Script

Script Name	Path Location	Purpose
DVS_run_validation_report.sql	$RBC_HOME/db/data_validation_system/bin	To execute the validation report. Run-time parameters need to be entered prior to running the program.
DVS_run_val_upb_report.sql	$RBC_HOME/db/data_validation_system/bin	To execute the instrument-level UPB validation report. Run-time parameters need to be entered prior to running the program.

Table 2-7: Data Validation Report Output Script

Script Name	Path Location	Purpose
DVS_run_val_report_example.txt	$RBC_HOME/db/data_validation_system/output	To demonstrate a typical data validation report based on executing the *DVS_run_val_report_example.txt* script using data other than the stylized data provided. (The stylized data does not contain any data validation violations.) Note: the report provided may not be identical to reports generated from data other than the stylized data provided due to technical environmental differences and parameter specifications.

2.4.3 RBCSIM Programs, and Build, Distribution, and Runtime Scripts

Table 2-2: RBCSIM Programs and Scripts

Script/Program Name	Path Location	Purpose
rbc_build.bash	$RBC_BIN/src/build/bin	Compile source files into binaries.
rbc_install.bash	$RBC_BIN/src/build/bin	Build and Install runtime components.
rbc_execute.bash	$RBC_HOME/bin	Execute entire RBCSIM run.
rbc_mrs	$RBC_HOME/bin	The Mortgage Related Securities (MRS) module projects cash flows for single class MBS, Mortgage Revenue Bonds (MRBs) and multi-class MBS (e.g. REMICs, Strips, etc.).
rbc_nmi	$RBC_HOME/bin	The Non-Mortgage Instruments module projects cash flows for debt instruments, non-mortgage related investments, guaranteed investment contracts, preferred stock, and derivative contracts.
Rdm	$RBC_HOME/bin	The Reporting and Decisions Module determines new debt issuance and investments, computing capital distributions, calculating operating expenses and taxes, creating pro forma balance sheets and income statements described in section 3.10 Operations, Taxes, and Accounting of the Risk-Based Capital Regulation, and the capital requirement.
wlcf	$RBC_HOME/bin	The Whole Loan Cash Flow module projects cash flows for retained and sold whole loans as well as for commitments.

Table 2-3: Configuration Files

File Name	Path Location	Purpose
rbcenv.csh	$RBC_HOME/config (deployment path) $RBC_HOME/src/build/config (src directory)	RBCSIM environment script for cshell. **Note**: The version in the source directory contains tags which are replaced during the installation process.
rbcenv.bash	$RBC_HOME/config (deployment path) $RBC_HOME/src/build/config (src directory)	RBCSIM environment script for bash, Bourne, and Korn shells. **Note**: The version in the source directory contains tags which are replaced during the installation process.
environment.config	$RBC_HOME/config (deployment path) $RBC_HOME/src/build/config (src directory)	Runtime configuration file.
acme_up.config	$RBC_HOME/config (deployment path) $RBC_HOME/src/build/config (src directory)	Acme, up scenario configuration file. **Note**: The version in the source directory contains tags which are replaced during the installation process.
acme_dn.config	$RBC_HOME/config (deployment path) $RBC_HOME/src/build/config (src directory)	Acme, down scenario configuration file. **Note**: The version in the source directory contains tags which are replaced during the installation process.
Common.mak	$RBC_HOME/src/common/build	The Common.mak provides the environment configuration to perform an application build.

2.4.4 RBCSIM Source Code by Component

2.4.4.1 Whole Loan

Table 2-4: Whole Loan Source Files

File Name	Path Location	Purpose
wlcf.cpp	/whole_loan/src	Main function of the whole loan cash flow module. It is responsible for instantiating/destroying parameter, database, interest rate, property valuation, data input manager, data collector and aggregator manager objects. It initiates the user-defined number of threads and makes sure all the threads start properly.
wlcfAggregationBucket.h wlcfAggregationBucket.cpp	/whole_loan /inc /whole_loan /src	Class definition of aggregation bucket. This class accumulates the results of the whole loan cash flow calculation and stores them into a map data structure for later output to a file.
wlcfAggregator.h wlcfAggregator.cpp	/whole_loan /inc /whole_loan /src	Definition of class wlcfAggregator. This class holds a collection of class wlcfAggregationBucket according to their aggregation types. It will pass the right data from wlcfWLCF to appropriate wlcfAggregationBucket objects to perform aggregation functions based on predefined aggregation algorithms.
wlcfAggregatorMgr.h wlcfAggregatorMgr.cpp	/whole_loan /inc /whole_loan /src	Definition of class wlcfAggregatorMgr. This class is responsible for maintaining instances of wlcfAggregator. There is one and only one wlcfAggregator instance for each thread. When the calculation of the whole loan cashflows is finished, wlcfAggregatorMgr will merge all the data into a single data collection, and then write it out to the whole loan cash flow file.
wlcfArmRelDataElmt.h wlcfArmRelDataElmt.h	/whole_loan /inc /whole_loan /src	Class definition of arm related data elements. This class is a data map of the arm_related_data_elements and arm_cmt_data_elements tables. It provides the set/get methods for accessing the database columns.
wlcfCalc.h wlcfCalc.cpp	/whole_loan /inc /whole_loan /src	Class definition of WLCF calculation. This class is responsible for the calculation order of the steps for producing whole loan cash flows.
wlcfCE.h wlcfCE.cpp	/whole_loan /inc /whole_loan /src	Class definition of credit enhancement, which is inherited from wlcfGLS. This class performs the credit enhancement calculation for single family and multifamily loans with the supplied input data from wlcfInputData, wlcfMA, wlcfDP and wlcfGLS. It stores the output results in its member attributes for later calculation of net loss severity. See section 3.6.3.6.4 of the RBC rule.
wlcfCeDataElmt.h wlcfCeDataElmt.cpp	/whole_loan /inc /whole_loan /src	Class definition of ce_data data elements. This class is a data map of the ce_data_elements and ce_cmt_data_elements tables. It provides the set/get methods for accessing the database columns.
wlcfCMT.h wlcfCMT.cpp	/whole_loan /inc /whole_loan /src	Class definition of commitments, which is inherited from wlcfInputData. This class performs the Commitments calculation with the supplied input data from wlcfInputData and stores the output results in its member attributes for later calculation of mortgage amortization. See section 3.2 of the RBC rule.

File Name	Path Location	Purpose
wlcfConfig.h wlcfConfig.cpp	/ whole_loan /inc / whole_loan /src	Class definition of wlcfConfig. This class is responsible for validating all necessary WLCF parameter values from the Parameter module. It displays warning and error messages if the parameters are inconsistent. It may also terminate the program if the errors are severe.
wlcfDataCollector.h wlcfDataCollector.cpp	/ whole_loan /inc / whole_loan /src	Class definition of intermediate data collector. This class performs as a callback to support the register/unregister mechanism supporting data collection throughout the calculation of the whole loan cash flows.
wlcfDP.h wlcfDP.cpp	/ whole_loan /inc / whole_loan /src	Class definition of default and prepayments, which is inherited from wlcfMA. This class performs calculation of default and prepayments of single family and multifamily loans with the supplied input data from wlcfInputData and wlcfMA. It stores the output results in its member attributes for later calculation of gross loss severity. See section 3.6.3.4 and 3.6.3.5 of the RBC Rule.
wlcfGLS.h wlcfGLS.cpp	/ whole_loan /inc / whole_loan /src	Class definition of gross loss severity, which is inherited from wlcfDP. This class performs the calculation of gross loss severity for single family and multifamily loans with the supplied input data from wlcfInputData, wlcfMA and wlcfDP. It stores the output results in its member attributes for later calculation of credit enhancement. See section 3.6.3.6 of the RBC Rule.
wlcfInputData.h wlcfInputData.cpp	/ whole_loan /inc / whole_loan /src	Class definition of single instrument input data. This class contains all the member attributes necessary to hold the single instrument input data received from the wlcfInputMgr. These member attributes supply the input data for all of the whole loan cash flow calculations. It also contains a pointer to the irpvMgr object to get the interest rate and property valuation data, the pointer to the wlcfDataCollector object to capture intermediate values and the pointer to the wlcfAggregatorMgr object to capture the output results of the whole loan cash flow simulation.
wlcfInputMgr.h wlcfInputMgr.cpp	/ whole_loan /inc / whole_loan /src	Class definition of the whole loan input manager. This class is responsible for buffering data queried from the database based on the reporting date and submitting entity id. It is also is responsible for instantiating/disposing the vector and map data structures of wlcfWLMaster, wlcfArmRelDataElmt, wlcfSfDataElmt, wlcfMfDataElmt and wlcfCeDataElmt.
wlcfMA.h wlcfMA.cpp	/ whole_loan /inc / whole_loan /src	Class definition of mortgage amortization, which is inherited from wlcfCMT. This class performs the mortgage amortization calculation for single family and multifamily loans with the supplied input data from wlcfInputData and wlcfCMT. It stores the output results in its member attributes for later calculation of default and prepayment rates. See section 3.6.3.3 of the RBC rule.
wlcfMfDataElmt.h wlcfMfDataElmt.cpp	/ whole_loan /inc / whole_loan /src	Class definition of multifamily data elements. This class is a data map of the multifamily_data_elements table. It provides the set/get methods for accessing the database columns.
wlcfNLS.h wlcfNLS.cpp	/ whole_loan /inc / whole_loan /src	Class definition of net loss severity, which is inherited from wlcfCE. This class performs the net loss severity calculation for single family and multifamily loans with the supplied input data from wlcfInputData, wlcfMA, wlcfDP, wlcfGLS and wlcfCE. It stores the output results in its member attributes for later calculation of whole loan cash flow. See section 3.6.3.6.5 of the RBC Rule.
wlcfRbcIoTypes.h wlcfRbcIoTypes.cpp	/ whole_loan /inc / whole_loan /src	Class definition of all intermediate input/output enumeration data types. This class supports the wlcfDataCollector class.

File Name	Path Location	Purpose
wlcfSfDataElmt.h wlcfSfDataElmt.cpp	/ whole_loan /inc / whole_loan /src	Class definition of single-family data elements. This class is a data map of the single_family_data_elements and sf_cmt_data_elements tables. It provides the set/get methods for accessing the database columns.
wlcfWLCF.h wlcfWLCF.cpp	/ whole_loan /inc / whole_loan /src	Class definition of whole loan cash flow, which is inherited from wlcfNLS. This class performs the whole loan cash flow calculation for single family and multifamily loans with the supplied input data from wlcfInputData, wlcfMA, wlcfDP, wlcfGLS, wlcfCE and wlcfNLS. It stores the output results in its member attributes for later feeding into wlcfAggregator. See section 3.6.3.7 of the RBC Rule.
wlcfWLMaster.h wlcfWLMaster.cpp	/ whole_loan /inc / whole_loan /src	Class definition of whole loan master. This class is a data map of the whole_loan_master and wl_cmt_master tables. It provides the set/get methods for accessing the database columns.

2.4.4.2 MRS

Table 2-5: MRS Source Files

File Name	Path Location	Purpose
AdjSecurity.cpp AdjSecurity.h	/mrs/src /mrs/inc	Implementation class for the amortization of adjustable rate MBS. See section 3.7.3.1 [b] of the RBC Rule.
Amortization.cpp Amortization.h	/mrs/src /mrs/inc	Parent class for the amortization of security collateral. See section 3.7.3.1 [b] of the RBC Rule.
cmoSubr.cpp cmoSubr.h	/mrs/src /mrs/inc	User defined C functions that are provided to the Intex CMO API for calculating projected cash flows for REMICs and Strips. See section 3.7.3.2 of the RBC Rule.
FixedSecurity.cpp FixedSecurity.h	/mrs/src /mrs/inc	Implementation class for the amortization of fixed rate MBS. See section 3.7.3.1 [b] of the RBC Rule.
IntRtIdxDefs.h	/mrs/inc	Defines a namespace for interest rate index names.
Mbs.cpp Mbs.h	/mrs/src /mrs/inc	Projects cash flows for Single Class Mortgage Backed Securities. See section 3.7.3.1 of the RBC Rule.
MbsCashFlow.cpp MbsCashFlow.h	/mrs/src /mrs/inc	Implements specialized cash flow processing and formatting functionality for Single-Class MBS. See section 3.7.3.1 [e] of the RBC Rule.
MbsDefPrepay.cpp MbsDefPrepay.h	/mrs/src /mrs/inc	Generates projected termination rates. See sections 3.7.3.1 [c] and 3.7.3.2 [a] of the RBC Rule.
Mrb.cpp Mrb.h	/mrs/src /mrs/inc	Projects cash flows for Mortgage Revenue Bonds. See section 3.7.3.3 of the RBC Rule.
MRS.cpp MRS.h	/mrs/src /mrs/inc	Parent class for the 3 MRS types, MRB, Single Class MBS, and Multi-Class MBS.
mrsAggregationBucket.cpp mrsAggregationBucket.h	/mrs/src /mrs/inc	Provides a container for aggregating cash flows, which share common characteristics.
MrsCashFlow.cpp MrsCashFlow.h	/mrs/src /mrs/inc	Responsible for processing and formatting the final projected cash flows. Implements the common cash flow processing functionality for all MRS modules.
MRSdefs.h	/mrs/inc	Defines a namespace for the constant string definitions used within the MRS modules.
mrsDriver.cpp	/mrs/src	Establishes high-level flow of control for the MRS application.
MrsMgr.cpp MrsMgr.h	/mrs/src /mrs/inc	Controlling class for the MRS modeling logic. See section 3.7 of the RBC Rule.
rbcExplanatoryVariables.cpp rbcExplanatoryVariables.h	/mrs/src /mrs/inc	Static lookup table for coefficients used in the calculation of termination rates. See section 3.6.3.4.3.2 of the RBC Rule.
Remic.cpp Remic.h	/mrs/src /mrs/inc	Projects cash flows for REMICS and Strips. See section 3.7.3.2 of the RBC Rule.

2.4.4.3 RDM

Table 2-6: RDM Source Files

File Name	Path Location	Purpose
BatchMgr.hpp BatchMgr.inl BatchMgr.cpp	/rdm/Common/inc /rdm/Common/inl /rdm/Common/source	Loads configuration parameters and manages batch execution.
BoundError.hpp BoundError.cpp	/rdm/Common/inc /rdm/Common/source	Validates boundary conditions for requested indexes during program execution.
CommandArg.hpp CommandArg.inl CommandArg.cpp	/rdm/Common/inc /rdm/Common/inl /rdm/Common/source	Processes the command line arguments.
Profile.hpp Profile.cpp	/rdm/Common/inc /rdm/Common/source	Base class for the various RDM profiles which reflect the configuration parameters.
ProfileMgr.hpp ProfileMgr.cpp	/rdm/Common/inc /rdm/Common/source	Manages the RDM's run and report profiles.
RdmDefs.hpp	/rdm/Common/inc	Common definitions for RDM.
ReportError.hpp ReportError.cpp	/rdm/Common/inc /rdm/Common/source	Exception object containing SQR error messages.
ReportMgr.hpp ReportMgr.cpp	/rdm/Common/inc /rdm/Common/source	Invokes SQR run-time to generate PostScript report files.
ReportProfile.hpp ReportProfile.cpp	/rdm/Common/inc /rdm/Common/source	Retains parameters required for report generation
RunProfile.hpp RunProfile.cpp	/rdm/Common/inc /rdm/Common/source	Retains parameters required for financial statement generation.
TentativeDividends.hpp TentativeDividends.cpp	/rdm/Common/inc /rdm/Common/source	Initializes arrays to store dividend payments.
Thread.hpp Thread.cpp	/rdm/Common/inc /rdm/Common/source	Wrapper class which implements the SUN Solaris Thread utilities.
AdministrativeExpensesCashFlow.hpp AdministrativeExpensesCashFlow.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Collects administrative expense data from the monthly journals. Calculates the monthly expense amounts.
CapitalClassification.hpp CapitalClassification.inl CapitalClassification.cpp	/rdm/Decisions/inc /rdm/Decisions/inl /rdm/Decisions/source	Calculates total capital position, core capital position and assigns the appropriate capital classification.
CommonShareRepurchase.hpp CommonShareRepurchase.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Makes the decision to repurchase common shares, generates the appropriate journal entries, and adjusts the outstanding shares starting position value.

File Name	Path Location	Purpose
DebtMaturityStructure.hpp DebtMaturityStructure.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Manages the debt maturity and the repricing matrix.
Dividends.hpp Dividends.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Calculates initial capital position classification, GSE earnings trend, tentative stock dividend payment amounts, and final capital position classification. Records the quarterly dividend payments to the journal for the quarter's ending month.
EndOfPeriod.hpp EndOfPeriod.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Manages decisions, which are to occur after all other decisions, but before the calculation of the final financial statements.
Funding.hpp Funding.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Uses ratios of short term funding and long-term funding to identify the mix of funding to be issued if the GSE experiences a cash shortfall. Calculates the par value, coupon rate, debt issuance cost, interest accrual and interest payments for short term and long term debt issued to cover cash shortfall.
InterestRates.hpp InterestRates.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Collects interest rate values for short term maturity debt, long term maturity debt, and GSE debt spread
Liquidity.hpp Liquidity.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Calculates the par value, interest, discount, and fees associated with liquidity investments in U.S. Treasury bills when the GSE experiences a cash surplus.
TaxCalculator.hpp TaxCalculator.inl TaxCalculator.cpp	/rdm/Decisions/inc /rdm/Decisions/inl /rdm/Decisions/source	Calculates the income tax monthly accruals, quarterly estimated tax payments, and annual tax payments or refunds.
Taxes.hpp Taxes.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Calculates the monthly accrual and quarterly payments for months prior to the first month of the stress test.
UniqueMonthOneAccountingEntries.hpp UniqueMonthOneAccountingEntries.cpp	/rdm/Decisions/inc /rdm/Decisions/source	Journal entries associated with month one of the stress test.
AssetDebenture.hpp AssetDebenture.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for debentures held as assets.
AssetDiscountInstrument.hpp AssetDiscountInstrument.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for discounted financial instruments held as assets.
CashFlow.hpp CashFlow.cpp	/rdm/FI/inc /rdm/FI/source	Base class, which defines common attributes for cash flows.
FinancialInstrument.hpp FinancialInstrument.cpp	/rdm/FI/inc /rdm/FI/source	Base class for the various financial instruments defining common structure and processing.
FinancialInstrumentFactory.hpp FinancialInstrumentFactory.cpp	/rdm/FI/inc /rdm/FI/source	Identifies the class object for processing each type of financial instrument.
InterfaceMgr.hpp InterfaceMgr.cpp	/rdm/FI/inc /rdm/FI/source	Loads and books the data from all cash flow files.
LiabDebenture.hpp LiabDebenture.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for debentures held as liabilities.
LiabDiscountInstrument.hpp LiabDiscountInstrument.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for discounted financial instruments held as liabilities.

File Name	Path Location	Purpose
MortgageSecurity.hpp MortgageSecurity.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for mortgage securities.
NMI_CashFlow.hpp NMI_CashFlow.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for NMI type financial instruments.
PreferredDividend.hpp PreferredDividend.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for preferred dividend type financial instruments.
RetainedMortgage.hpp RetainedMortgage.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for government insured mortgages, conventional single-family whole loans, and conventional multi-family whole loans in the retained portfolio.
RM_CashFlow.hpp RM_CashFlow.cpp	/rdm/FI/inc /rdm/FI/source	Reads the projected cash flow data for retained mortgage portfolio financial instruments.
SM_CashFlow.hpp SM_CashFlow.cpp	/rdm/FI/inc /rdm/FI/source	Reads the projected cash flow data for sold mortgage portfolio financial instruments.
SoldMortgage.hpp SoldMortgage.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for 30-year fixed-rate single-family whole loans, 20 year fixed-rate single-family whole loans, 15 year fixed-rate single-family whole loans, adjustable-rate single-family whole loans, balloon/reset single-family whole loans, other single-family whole loans, and multi-family whole loans in the sold mortgage portfolio.
SubordinatedDebt.hpp SubordinatedDebt.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for subordinated debt.
Swap.hpp Swap.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for swaps.
SwaptionCashSettlement.hpp SwaptionCashSettlement.cpp	/rdm/FI/inc /rdm/FI/source	Processes the projected cash flow data for cash settlement swaptions.
AdministrativeExpenses.hpp AdministrativeExpenses.inl AdministrativeExpenses.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the administrative expense information for Schedule L – Administrative Expenses from the monthly journals and saves administrative_expenses table data to the database.
AnalysisReport.hpp AnalysisReport.inl AnalysisReport.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the earnings ratios, yields and costs, credit for loan losses, and other ratios for the Analytics Report.
Assets.hpp Assets.inl Assets.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the retained portfolio, non-mortgage investments, cash, accrued interest receivable, foreclosed property net and other assets information for the balance sheet from the monthly journals and saves the asset data to the database.
BalanceSheet.hpp BalanceSheet.inl BalanceSheet.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the balance sheet line item data from the assets, liabilities, stockholder's equity, sold mortgage portfolio, derivative and capital position objects and saves the balance_sheet table data to the database.
CapitalStatement.hpp CapitalStatement.inl CapitalStatement.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the dividends and capital adequacy information to calculate and report the final Risk Based Capital classification

File Name	Path Location	Purpose
CashFinancing.hpp CashFinancing.inl CashFinancing.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the cash flow statement data for financing activities from the monthly journals and calculates the total cash from operating activities. Saves data to the database.
CashflowStatement.hpp CashflowStatement.inl CashflowStatement.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the cash flow statement information from the CashOperating, CashInvesting, and CashFinancing objects. Collects the cash position data from the monthly journals and calculates the monthly change in cash position. Saves data to the database.
CashInvesting.hpp CashInvesting.inl CashInvesting.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the cash flow statement data for investing activities from the monthly journals and calculates the total cash from investing activities. Saves data to the database.
CashOperating.hpp CashOperating.inl CashOperating.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the cash flow statement data for operating activities from the monthly journals and calculates the total cash from operating activities.
CreditForLoanLosses.hpp CreditForLoanLosses.inl CreditForLoanLosses.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the income statement loss information for retained and sold mortgage loans from the monthly journals and saves the data to the database.
DebtInstruments.hpp DebtInstruments.inl DebtInstruments.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the debt instrument expenses for Schedule I – Interest on Debt Securities from the monthly journals and saves the data to the database.
DebtSecurities.hpp DebtSecurities.cpp	/rdm/FS/inc /rdm/FS/source	Collects the debt instrument account balances for Schedule D – Debt Securities and Schedule N – Debt Portfolio by Maturity by Coupon. Saves the data to the database.
Derivatives.hpp Derivatives.inl Derivatives.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the interest rate swaps, caps, floors, corridors, etc. data from the monthly journals and saves the data to the database.
DerivativesWrap.hpp DerivativesWrap.inl DerivativesWrap.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Wraps all types of derivatives.
EarningRatios.hpp EarningRatios.inl EarningRatios.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the data and calculates the earnings ratios for the analytics report from monthly journals. Saves the data to the database.
FinancialStatement.hpp FinancialStatement.inl FinancialStatement.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects data for financial statements and sets the appropriate month. The Schedule K and Schedule M provision calculations are treated uniquely within this class.
FinancialStatementMgr.hpp FinancialStatementMgr.inl FinancialStatementMgr.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Manages the financial statement generation.
FinancialStatementsKeyInfo.hpp FinancialStatementsKeyInfo.inl FinancialStatemenstKeyInfo.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Obsolete.

File Name	Path Location	Purpose
GuaranteeFee.hpp GuaranteeFee.inl GuaranteeFee.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects guarantee fee data from the monthly journals and saves the data to the database
IncomeOnNonMortgageInvestments.hpp IncomeOnNonMortgageInvestments.inl IncomeOnNonMortgageInvestments.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects non-mortgage investment data from the monthly journals. Saves the data to the database.
IncomeStatement.hpp IncomeStatement.inl IncomeStatement.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the Income Statement line item data for income (i.e., net interest and guarantee fee income) expenses (i.e., administrative, mortgage loss, income tax) and stock dividends from the monthly journals. Calculates net income both before and after taxes. Saves the data to the database.
IncomeTaxes.hpp IncomeTaxes.inl IncomeTaxes.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the income tax provision, amount paid or refundable, valuation adjustment, and account balance from the monthly journals for Schedule M – Income Taxes. Saves the data to the database.
InterestIncome.hpp InterestIncome.inl InterestIncome.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the interest income and net amortized balance amounts from the monthly journals for Schedule G, G1, G2 – Income from Retained Mortgages. Saves the data to the database.
InerestOnDebtSecurities.hpp InerestOnDebtSecurities.inl InerestOnDebtSecurities.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the interest on debt securities from the monthly journals for Schedule I – Interest on Debt Securities. Saves the data to the database.
InterestRateSwaps.hpp InterestRateSwaps.inl InterestRateSwaps.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects interest rate swap data from the monthly journals for Schedule O – Derivatives. Saves the data to the database.
Investments.hpp Investments.inl Investments.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects non-mortgage investment data for Schedule B – Investments and Schedule H – Income on Investment Securities from the monthly journals. Saves the data to the database.
Liabilities.hpp Liabilities.inl Liabilities.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects debt security (i.e. new securities, existing securities and total securities) data from the monthly journals. Also, calculates accrued interest payable for debt securities. Saves the data to the database.
LossPortfolio.hpp LossPortfolio.inl LossPortfolio.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects retained and sold mortgage portfolio loss data (i.e., account balance, provision for year, charge-offs for the year) for Schedule K – Allowance for Losses from the monthly journals. Saves the data to the database.
MortgageLossPortfolio.hpp MortgageLossPortfolio.inl MortgageLossPortfolio.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects provision for mortgage losses from the monthly journals and calculates projected losses. Saves the data to the database.
MortgagePortfolioGuranteeFee.hpp MortgagePortfolioGuranteeFee.inl MortgagePortfolioGuranteeFee.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects mortgage portfolio (single and multi family) guarantee fees for Schedule G, G1 and G2 from the monthly journals. Saves the data to the database.
MortgagePortfolioInterestIncome.hpp MortgagePortfolioInterestIncome.inl MortgagePortfolioInterestIncome.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects mortgage portfolio (single- and multifamily) interest income for Schedule G, G1 and G2 from the monthly journals. Saves the data to the database.

File Name	Path Location	Purpose
MortgagePortfolioRetained.hpp MortgagePortfolioRetained.inl MortgagePortfolioRetained.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects retained mortgage portfolio (single- and multifamily) unpaid principal balances, net premiums or discounts and reserve for losses for Schedule A1 and A2 from the monthly journals. Saves the data to the database.
MortgagePortfolioSold.hpp MortgagePortfolioSold.inl MortgagePortfolioSold.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects sold mortgage portfolios (single- and multifamily) unpaid principal balance for Schedules F1 and F2 from the monthly journals. Saves the data to the database.
MortgageProductTypes.hpp MortgageProductTypes.inl MortgageProductTypes.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Defines the mortgage product types (e.g., fixed 30 year, adjustable rate, balloon, etc.) for Schedules G1, G2, J1, and J2. Saves the data to the database.
MultiFamily.hpp MultiFamily.inl MultiFamily.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects multifamily mortgage data from the monthly journals. Saves the data to the database.
MultiFamilyLossPortfolio.hpp MultiFamilyLossPortfolio.inl MultiFamilyLossPortfolio.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects multifamily mortgage portfolio loss data (i.e., account balance, provision for year, charge-offs for the year) from the monthly journals. Saves the data to the database.
NonMortgageInvestments.hpp NonMortgageInvestments.inl NonMortgageInvestments.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects non-mortgage investment data (including investment linked derivatives) from the monthly journals. Saves the data to the database.
OtherAssets.hpp OtherAssets.inl OtherAssets.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects other asset data for the balance sheet and Schedule C – Other Assets from the monthly journals. Saves the data to the database.
OtherLiabilities.hpp OtherLiabilities.inl OtherLiabilities.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects other liabilities for the balance sheet and Schedule E – Other Liabilities from the monthly journals. Saves the data to the database.
OtherRatios.hpp OtherRatios.inl OtherRatios.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects financial and other ratios for the Analytics Report from the monthly journals. Saves the data to the database.
PeriodicFinancialStatement.hpp PeriodicFinancialStatement.inl PeriodicFinancialStatement.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Performs basic actions on the periodic financial statements (i.e., balance sheet, income statement, statement of cash flows), such as save, bringForward, loadFromJournal, calculateTotal, etc.
ProcessMgr.hpp ProcessMgr.inl ProcessMgr.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Controls the execution of the RDM using the manager objects (i.e., InterfaceMgr, FinancialStatementMgr, etc.). This class is instantiated by the main function or the BatchMgr. Provides application initialization as well as shut down capabilities for error handling. Spawns the threads for the generation of the financial statements.
ProvisionForMortgageLosses.hpp ProvisionForMortgageLosses.inl ProvisionForMortgageLosses.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Calculates the provision for mortgage losses for the retained and sold mortgage portfolios.

File Name	Path Location	Purpose
ProvisionForMortgageLossesWrap.hpp ProvisionForMortgageLossesWrap.inl ProvisionForMortgageLossesWrap.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Calculates totals of single- and multifamily retained and sold portfolio, and saves to the loss_portfolio database table.
SingleFamilyLossPortfolio.hpp SingleFamilyLossPortfolio.inl SingleFamilyLossPortfolio.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects single-family mortgage portfolio loss data from the monthly journals. Saves the data to the database.
SingleFamilyMortgageData.hpp SingleFamilyMortgageData.inl SingleFamilyMortgageData.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects single-family mortgage data from the monthly journals. Saves the data to the database.
StartingPosition.hpp StartingPosition.cpp	/rdm/FS/inc /rdm/FS/source	Creates journal entries for the starting position data read in from the database.
StockHolderEquity.hpp StockHolderEquity.inl StockHolderEquity.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects the stockholder's equity accounts data for the balance sheet from the monthly journals. Saves the data to the database.
YieldsAndCosts.hpp YieldsAndCosts.inl YieldsAndCosts.cpp	/rdm/FS/inc /rdm/FS/inl /rdm/FS/source	Collects data for the yields and costs section of the Analytics Report from the monthly journals. Saves the data to the database.
CashFileInfo.hpp CashFileInfo.cpp	/rdm/GUI/source /rdm/GUI/source	Establishes the location and names for cashflow and starting position files.
FrcmMain.cpp	/rdm/GUI/source	Contains the main function for the RDM.
AssetEntry.hpp AssetEntry.cpp	/rdm/Journal/inc /rdm/Journal/cpp	Defines the debit and credit logic for asset accounts.
CapitalEntry.hpp CapitalEntry.cpp	/rdm/Journal/inc /rdm/Journal/cpp	Defines the debit and credit logic for capital accounts.
ExpenseEntry.hpp ExpenseEntry.cpp	/rdm/Journal/inc /rdm/Journal/cpp	Defines the debit and credit logic for expense accounts.
Factory.hpp Factory.cpp	/rdm/Journal/inc /rdm/Journal/cpp	Defines the entry type for each journal entry (i.e., asset, liability, revenue, expense, capital, off-balance sheet accounts).
Journal.hpp Journal.inl Journal.cpp	/rdm/Journal/inc /rdm/Journal/inl /rdm/Journal/cpp	Container for 121 MonthlyJournal objects (121 months: 1 for the base month and 120 months for the 10 year stress period). Defines Journal assignment operations and initialization of Journal indices for the simulation.
JournalEntry.hpp JournalEntry.inl JournalEntry.cpp	/rdm/Journal/inc /rdm/Journal/inl /rdm/Journal/cpp	Defines the process of booking transactions into a journal..
JournalMgr.hpp JournalMgr.inl JournalMgr.cpp	/rdm/Journal/inc /rdm/Journal/inl /rdm/Journal/cpp	Instantiates the Journal object. Mechanism by which the other RDM objects interface with the Journal object for booking journal entries.
LiabilityEntry.hpp LiabilityEntry.cpp	/rdm/Journal/inc /rdm/Journal/cpp	Defines the debit and credit logic on liability accounts.

File Name	Path Location	Purpose
MonthlyJournal.hpp MonthlyJournal.inl MonthlyJournal.cpp	/rdm/Journal/inc /rdm/Journal/inl /rdm/Journal/cpp	Stores the monthly changes to the general ledger in a keyed collection of JournalEntry objects.
OffBalanceSheetAssetEntry.hpp OffBalanceSheetAssetEntry.cpp	/rdm/Journal/inc /rdm/Journal/cpp	Defines the debit and credit logic for off balance sheet asset accounts.
RevenueEntry.hpp RevenueEntry.cpp	/rdm/Journal/inc /rdm/Journal/cpp	Defines the debit and credit logic for revenue accounts.
AnalysisReport.sqr AnalysisReportAscii.sqr AnalysisReportCommon.sqr BalanceSheet.sqr BalanceSheetAscii.sqr BalanceSheetCommon.sqr CapitalAdequacy.sqr CapitalAdequacyAscii.sqr CapitalAdequacyCommon.sqr Cashflow.sqr CashflowAscii.sqr CashflowCommon.sqr ComDiv.sqr Common.sqr CommonAscii.sqr Dividends.sqr IncomeStatement.sqr IncomeStatementAscii.sqr IncomeStatementCommon.sqr OFHEO.sqr RunProfile.sqr ScheduleA1.sqr ScheduleA1Ascii.sqr ScheduleA1Common.sqr ScheduleA2.sqr ScheduleA2Ascii.sqr ScheduleA2Common.sqr ScheduleB.sqr ScheduleBAscii.sqr ScheduleBCommon.sqr ScheduleC.sqr ScheduleCAscii.sqr ScheduleCCommon.sqr	/rdm/SQR/source	SQR definition format files for the RDM financial reports.

File Name	Path Location	Purpose
ScheduleD.sqr		
ScheduleD1.sqr		
ScheduleD1Ascii.sqr		
ScheduleD2.sqr		
ScheduleD2Ascii.sqr		
ScheduleDAscii.sqr		
ScheduleDCommon.sqr		
ScheduleE.sqr		
ScheduleEAscii.sqr		
ScheduleECommon.sqr		
ScheduleF1.sqr		
ScheduleF1Ascii.sqr		
ScheduleF1Common.sqr		
ScheduleF2.sqr		
ScheduleF2Ascii.sqr		
ScheduleF2Common.sqr		
ScheduleG.sqr		
ScheduleG1.sqr		
ScheduleG1Ascii.sqr		
ScheduleG2.sqr		
ScheduleG2Ascii.sqr		
ScheduleGAscii.sqr		
ScheduleGCommon.sqr		
ScheduleH.sqr		
ScheduleHAscii.sqr		
ScheduleHCommon.sqr		
ScheduleI.sqr		
ScheduleIAscii.sqr		
ScheduleICommon.sqr		
ScheduleJ.sqr		
ScheduleJ1.sqr		
ScheduleJ1Ascii.sqr		
ScheduleJ2.sqr		
ScheduleJ2Ascii.sqr		
ScheduleJAscii.sqr		
ScheduleJCommon.sqr		
ScheduleK.sqr		
ScheduleKAscii.sqr		
ScheduleKCommon.sqr		
ScheduleL.sqr		

File Name	Path Location	Purpose
ScheduleLAscii.sqr		
ScheduleLCommon.sqr		
ScheduleM.sqr		
ScheduleMAscii.sqr		
ScheduleMCommon.sqr		
ScheduleN.sqr		
ScheduleN1.sqr		
ScheduleN2.sqr		
ScheduleNCommon.sqr		
ScheduleO.sqr		
ScheduleOAscii.sqr		
ScheduleOCommon.sqr		

2.4.4.4 NMI

Table 2-7: NMI Source Files

File Name	Path Location	Purpose
amt_debt.h amt_debt.cc	/nmi/inc /nmi/src	The Main driver for processing Alternative Modeling Treatment (AMT) data and producing AMT cashflow files as described in section 3.9 of the RBC Rule.
bond_cash.h bond_cash.cc	/nmi/inc /nmi/src	Provides methods for processing cashflows for bond instruments and the structures for storing and generating bond instrument cashflow files.
Cmocashflow.h	/nmi/inc	Provides the structures for storing cashflow-related data generated by the Intex library of routines.
cmosubr.h cmosubr.cc	/nmi/inc /nmi/src	The set of "C" functions or callbacks that are passed as parameters into the Intex library routines. These routines are used primarily when calculating the Principal Factor Amount at each payment date for Asset-Backed Securities. See section 3.8.3.3 of the RBC Rule.
corp_debt.h corp_debt.cc	/nmi/inc /nmi/src	Contains the majority of the business logic for processing debt instruments and a portion of the Day Count implementation.
cusip_intex.h cusip_intex.cc	/nmi/inc /nmi/src	Provides methods for mapping from Cusip to the Intex Deal identifier.
date.h date.cc	/nmi/inc /nmi/src	Provides the structures for storing date-related data and a set of methods for manipulating them.
day_count.h day_count.cc	/nmi/inc /nmi/src	Provides the structures for storing Day Count data and a set of methods for manipulating them. Specifically, provides methods for determining the coupon factor to use when calculating an instrument's interest payment.
debt_coupon.h debt_coupon.cc	/nmi/inc /nmi/src	Contains the majority of the business logic for processing a debt instrument's coupon-related data.
definedate.h	/nmi/inc	Provides a set of date related mappings such as "JAN" -> "January" used by the date processing methods.
enum.h	/nmi/inc	Provides a set of enumerations related to the "definedata.h" mappings.
error.h error.cc	/nmi/inc /nmi/src	Provides a set of routines for handling errors and generating error messages when running the NMI modules.
file_gen.h file_gen.cc	/nmi/inc /nmi/src	Provides the necessary routines for producing cashflow files from a set of cashflow-related structures and data. See section 3.8.4 of the RBC Rule. Also provides the methods for applying counterparty haircuts to an instrument.
finance.h finance.cc	/nmi/inc /nmi/src	Provides a set of methods that implement common Financial functions, such as MacCauley Duration, Modified Duration, Convexity, etc.
fix_coupon.h fix_coupon.cc	/nmi/inc /nmi/src	Provides the majority of the business logic for processing a fixed rate instrument's coupon-related data.
fixincome.h fixincome.cc	/nmi/inc /nmi/src	The Main driver for processing Non-Mortgage Instrument (NMI) data - loads the NMI data from the database, performs preprocessing of the data and passes the data on to other classes.

File Name	Path Location	Purpose
		Also provides the majority of the business logic for processing SWAPs and SWAPTIONs.
float_coupon.h float_coupon.cc	/nmi/inc /nmi/src	Provides the majority of the business logic for processing a floating rate instrument's coupon-related data.
fut_opt.h fut_opt.cc	/nmi/inc /nmi/src	Main driver for processing future's data and producing future's cash flows.
fut_opt_utl.h fut_opt_utl.cc	/nmi/inc /nmi/src	Provides a set of utility routines used for processing Future-related cashflows.
gse.h	/nmi/inc	Provides the set of structures for storing GSE data, loaded from the database, for processing.
gse_code.h gse_code.cc	/nmi/inc /nmi/src	Provides the structures and methods for mapping GSE-related data to/from a String or an element of an enumerated type. Similar to a Map data structure but specific to the GSE's quarterly data submission.
gse_data.cc	/nmi/src	Provides methods for displaying/printing the structures provided by the "gse_code" maps.
gse_bind.cc	/nmi/src	Provides the methods that bind or map the GSE data from the database into the structures provided by "gse.h".
gse_org_data.h	/nmi/inc	Provides additional, intermediate data structures that support the processing of GSE-related data and the production of cashlfow files.
gse_syb_define.h	/nmi/inc	Provides the set of "Sybase" specific structures for loading GSE data from the database. Used in conjuction with "gse.h", gse_bind.cc" and "gsectlib.cc".
gse_syb_table.h	/nmi/inc	Defines the "signatures" of a handful of common Sybase-related routines.
gsectlib.h gsectlib.cc	/nmi/inc /nmi/src	Provides the set of methods for reading/loading data from the database to be used by the model for processing and producing cashflow files. Serves as an abstract interface for calling Sybase's database API.
index_store.h index_store.cc	/nmi/inc	Provides the structures for storing Interest Rate data and the methods for writing and reading that data.
intex.h intex.cc	/nmi/inc /nmi/src	Provides the set of structures and methods for calling the Intex routines. Serves as an abstract interface for calling Intex's CMO API.
intex_cashflow.h intex_cashflow.cc	/nmi/inc /nmi/src	Provides the structures for storing the results from Intex API routines and the set of methods for using those structures. Also provides the methods for processing the Intex-produced data's Interest and Principal payment values.
junk_store.h junk_store.cc	/nmi/inc /nmi/src	Provides a set of structures and methods for printing and displaying NMI-processed data.
lia_enum.h	/nmi/inc	Provides a set of enumerated data types specific to the model.
liab_choice.h liab_choice.cc	/nmi/inc /nmi/src	Provides a set of methods for processing command-line arguments and configuration file parameters.
math.cc	/nmi/src	Provides a set of methods that implement common mathematical functions, such as linear interpolation, extrapolation, min/max of

File Name	Path Location	Purpose
		floating values, etc.
NMIParameterKeys.h	/nmi/inc	Defines configuration file parameters specific to the NMI.
option_eval.h option_eval.cc	/nmi/inc /nmi/src	Provides a set of methods for processing Option-related data. Methods include determining an Option's exercise data, strike price, equivalent yield, payment frequency, etc.
prin_enum.h	/nmi/inc	Provides an enumeration of Principal types.
remic_portfolio.h	/nmi/inc	Provides the structure for creating, storing and manipulating a REMIC portfolio.
step_coupon.h step_coupon.cc	/nmi/inc /nmi/src	Provides the majority of the business logic for processing a step-rate instrument's coupon-related data.
strroutine.cc	/nmi/src	Provides a set of methods for manipulating string-related data.
syb_common.h syb_common.cc	/nmi/inc /nmi/src	Provides a set of "defines" and "macros" common to all methods that access the Sybase's database API.
syb_set_col.cc	/nmi/src	Provides a set of structures that store meta-data, based on data type. This meta-data is populated during calls to the Sybase database API.
unamort.h unamort.cc	/nmi/inc /nmi/src	Provides a set of methods for performing amortization of deferred, premiums, discounts and fees.

2.4.4.5 Params

These files are no longer used. Reference the common modules.

Table 2-8: Params Source Files

File Name	Path Location	Purpose
ParameterKeys.h ParameterKeys.cpp	/params/inc /params/src	Contains the Parameter Keys used by the application.
Parameters.h Parameters.cpp	/params/inc /params/src	Provides the methodology for parsing a configuration file, creating the internal representation of key/value pair mappings, associating a domain with specific parameter values, accessing parameter values based on key and/or domain and creating and obtaining the Singleton instance.
ParametersFileLoader.h ParametersFileLoader.cpp	/params/inc /params/src	Provides the methodology for processing the command-line arguments and loading a configuration file for processing by the Parameters' methods.
TestInstrID.cpp	/params/src	Test driver for loading the set of Instrument IDs to process from a file.
TestParameters.cpp	/params/src	Test driver for Parameters methods.
TestParametersFileLoader.cpp	/params/src	Test driver for the ParametersFileLoader methods.

2.4.4.6 DBMGR

Table 2-9: DBMGR Source Files

File Name	Path Location	Purpose
DbRecord.h DbRecord.cpp	/dbmgr/inc /dbmgr/source	A vector of string values which represent a single row of SQL query results.
DbRecordSet.h DbRecordSet.cpp	/dbmgr/inc /dbmgr/source	A vector of DbRecord values which represent all rows returned by an SQL query.
DbToken.h DbToken.cpp	/dbmgr/inc /dbmgr/source	Set of name/value pair lists used by the database manager object to construct an SQL command.
FSError.hpp FSError.cpp	/dbmgr/inc /dbmgr/source	Obsolete.
ImplDbMgr.h ImplDbMgr.cpp	/dbmgr/inc /dbmgr/source	Implementation of the class that handles the Sybase database connection.
InterfaceDbMgr.h InterfaceDbMgr.cpp	/dbmgr/inc /dbmgr/source	Interface class, which controls interactions between the application and the database manager implementation class.

2.4.4.7 Common

Table 2-10: Common Source Files

File Name	Path Location	Purpose
comRbcEnum.h comRbcEnum.cpp	/common/inc /common/src	Defines all the enumeration types in whole_loan_master, wl_master_cmt, arm_related_data_element, arm_cmt_data_elements, single_family_data_elements, sf_cmt_data_elements, multifamily_data_elements, ce_data_elements and ce_cmt_data_elements tables. Declares function prototypes to convert various enumeration types to string and string arrays of various table data type.
comRbcIdxCodeDef.h comRbcIdxCodeDef.cpp	/common/inc /common/src	Defines all of the available index codes used in the simulation by enumeration type. Declare function prototypes to convert enumeration type to string type.
comMutex.h comMutex.cpp	/common/inc /common/src	Class definition of mutual exclusive locking wrapper. This class is used to lock/unlock threads in order to avoid data collisions in the multithreaded environment.
comIRR.h comIRR.cpp	/common/inc /common/src	Class definition of the internal rate of return calculation. This class applies Newton's method and the bisection method to calculate the internal rate of return.
comCumNorm.h comCumNorm.cpp	/common/inc /common/src	Class definition of the cumulative normal distribution. This class calculates the cumulative normal distribution to support the calculation of single-family default and prepayment rates.
Encryption.h Encryption.cpp	/common/inc /common/src	Class definition of the simple encryption algorithm used for storing the user's password.
encryption.cpp	/common/src	Main driver module for the Encryption class. The resulting executable encrypts and decrypts passwords for user scripts.
EnvSubs.h EnvSubs.cpp	/common/inc /common/src	Class used by Param_IniFile to substitute environment variables placed within the value portion of the key-value pair. The format is ${VARIABLE_NAME}.
LogMsg.h LogMsg.cpp	/common/inc /common/src	Class used by the application threads to output log results. Each thread obtains a dedicated LogMsg object from the Logger.
Logger.h Logger.cpp	/common/inc /common/src	Singleton (builder) class used as the log controller. Logger creates, distributes, and manages LogMsg objects as needed by the application threads.
Message.h Message.cpp	/common/inc /common/src	Class used to format warning and error messages.
MessageTable.h MessageTable.cpp	/common/inc /common/src	Abstract container class to manage error codes.
Param_IniFile.h Param_IniFile.cpp	/common/inc /common/src	Instance of Parameters class used to implement configuration options for the RBC modules. An example can be found in Appendix B of the RBC User Manual.
Parameters.h Parameters.cpp	/common/inc /common/src	Abstract class defining the Parameters API. Consists of a vector of ParamSections.
ParamHandle.h ParamHandle.cpp	/common/inc /common/src	A singleton class used by the various source modules to access the primary configuration file.

File Name	Path Location	Purpose
ParamKeys.h ParamKeys.cpp	/common/inc /common/src	The lowest level component used by Parameters. This class consists of a key-value pair.
ParamSection.h ParamSection.cpp	/common/inc /common/src	This class defines a Parameters group and consists of a vector of ParamKeys.
RBC_MessageTable.h RBC_MessageTable.cpp	/common/inc /common/src	Set of warning and error messages used by the RBC modules.
SigHandler.h SigHandler.cpp	/common/inc /common/src	Singleton class used to handle run-time exceptions.
TimeDate.h TimeDate.cpp	/common/inc /common/src	Class definition used for time and date formatting.

2.4.4.8 IR_PV

Table 2-11: IR_PV Source Files

File Name	Path Location	Purpose
irpvIntrRate.h irpvIntrRate.cpp	/ir_pv/inc /ir_pv/src	Class definition of Interest Rate calculation/query. It holds the map and vector data structures that store the projected interest rates, which are calculated based on historical interest rate data.
irpvPV.h irpvPV.cpp	/ir_pv/inc /ir_pv/src	Class definition of the property valuation module. It holds the vector data structures to store the results of the projected house prices.
irpvMgr.h irpvMgr.cpp	/ir_pv/inc /ir_pv/src	Class definition of interface to the interest rate and property valuation modules. This class responds to user supplied parameters and instantiates the irpvIntrRate and the irpvPV objects. It also acts as a placeholder of interest rate index data, which can be accessed by other modules of RBC simulation model.
Irpv.cpp	/ir_pv/src	Unit tester for the interest rate module. It uses the parameters module.

2.4.5 Sample Data Files

Table 2-12: RBC Data File Samples

RBC – Related Data Tables Install Location = $RBC_INSTALL_ROOT/db/data/Acme(/lookup)		
File Name	**Data Table Map**	**Description**
amt.dat	amt	The Alternative Modeling Treatments table is used to store information from the other source tables where the information is incomplete. Data for this table is submitted quarterly, if relevant. Reference: RBC Report Instructions.
arm_cmt_data_elements.dat	arm_cmt_data_elements	The ARM –Commitments table contains information about adjustable rate mortgage loans for mortgage commitments. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
arm_related_data_elements.dat	arm_related_data_elements	The ARM table contains information about adjustable rate mortgage loans. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
ce_cmt_data_elements.dat	ce_cmt_data_elements	The Distinct Credit Enhancement Combination (DCC) - Commitments table contains information for each owned or guaranteed credit-enhanced mortgage commitment. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
ce_contract_elements.dat	ce_contract_elements	The Credit Enhancement Contracts table contains information for each contract that affects one or more owned or guaranteed loans. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
ce_data_elements.dat	ce_data_elements	The Distinct Credit Enhancement Combination (DCC) table contains information for each owned or guaranteed credit-enhanced mortgage. Data for this table is submitted quarterly. Reference: RBC Report Instructions. Reference: RBC Report Instructions.
cntrprty_cred_rating.dat	cntrprty_cred_rating	The Counterparty Credit Rating table contains credit rating information about the counterparty from an NRSRO (or otherwise approved by OFHEO) for each instrument comprising a derivative contract. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
cntrprty_entity_lkp.dat	cntrprty_entity_lkp	The Counterparty Entity Lookup table contains information about each counterparty or counterparty parent involved in a derivative contract with the Enterprise. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
financial_instmt_mstr.dat	financial_instmt_mstr	The Financial Instrument Master table contains information about each non-mortgage instrument other than futures and options on futures. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
fsm_adj_hpi.dat	fsm_adj_hpi	Stores projected hpi rates. Used only for analysis.
fsm_adj_rpi.dat	fsm_adj_rpi	Stores projected rpi rates. Used only for analysis.
fsm_benchmark_hpi.dat	fsm_benchmark_hpi	Lookup table for the House Price Index (HPI) series for the West South Central Census Division for the years 1984-1993. Source: RBC Rule, Table 3-19 Section 3.1.
fsm_benchmark_rpi.dat	fsm_benchmark_rpi	Lookup table for the population-weighted average of the monthly growth of the Rent of Primary Residence component of the Consumer Price Index-Urban which is generated by the U.S. Department of Commerce Bureau of Labor Statistics. Source: RBC Rule, Table 3-20 Section 3.1.
fsm_benchmark_vr.dat	fsm_benchmark_vr	Lookup table for the population-weighted average of annual rental vacancy rates from the U.S. Department of Commerce, Bureau of the Census' Housing Vacancy Survey. Source: RBC Rule, Table 3-20 Section 3.1.
fsm_hist_idx.dat	fsm_hist_idx	Lookup table for historical interest rates. Data is updated quarterly. Source: RBC Rule, Table 3-18 Section 3.1.

RBC – Related Data Tables

Install Location = $RBC_INSTALL_ROOT/db/data/Acme(/lookup)

File Name	Data Table Map	Description
fsm_lookup_aoltv.dat	fsm_lookup_aoltv	Lookup table for amortized original loan to value ratio (original LTV adjusted for the change in UPB but not for changes in property value). Source: RBC Rule, Table 3-59 Section 3.7.
fsm_lkp_cmt_ratio.dat	fsm_lookup_cmt_ratio	Lookup table for constant maturity treasury ratios to the Ten-Year CMT. Source: RBC Rule, Table 3-26 Section 3.3.
fsm_lkp_derv_haircut.dat	fsm_lookup_derivative_haircut	Lookup table for derivative haircuts. Source: RBC Rule, Table 3-31 Section 3.5.
fsm_lkp_idx.dat	fsm_lookup_idx	Lookup table used for validating index codes.
fsm_lkp_idx_assoc.dat	fsm_lookup_idx_assoc	Lookup table for a given index's associated treasury index. Source: RBC Rule, Table 3-27 Section 3.3.
fsm_lkp_natl_avg_hpi.dat	fsm_lookup_natl_avg_hpi	Lookup table for national average house price index. Data is updated quarterly. Note: there is a one-quarter lag. Source: RBC Rule, Table 3-60 Section 3.7.
fsm_lkp_non_derv_hair.dat	fsm_lookup_non_derivative_hair	Lookup table for non-derivative haircuts. Source: RBC Rule, Table 3-31 Section 3.5.
fsm_lkp_sf_def_pp_coeff.dat	fsm_lookup_sf_def_prepay_coeff	Lookup table for single family default prepay coefficients. Source: RBC Rule, Table 3-35 Section 3.6.
fsm_proj_idx.dat	fsm_proj_idx	Stores projected rates for a specific stress period. Rates will only be stored for analysis.
fsm_run_profile.dat	fsm_run_profile	Stores information on analysis data contained in the fsm_adj_hpi, fsm_adj_rpi, and fsm_proj_idx tables.
futures_options.dat	futures_options	The Futures Options table contains information about each futures contracts or options (put or calls) on futures contracts. Data for this table is submitted quarterly, if relevant. Reference: RBC Report Instructions.
idx_formula.dat	idx_formula	The Index Formula table contains information necessary to calculate interest payments on an instrument. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
instmt_assn.dat	instmt_assn	The Instrument Association table contains information that links two or more instruments comprising a derivative contract. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
instmt_cred_rating.dat	instmt_cred_rating	The Instrument Credit Rating table contains credit rating information from an NRSRO (or otherwise approved by OFHEO) about each instrument held as an asset. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
int_paymt_formula.dat	int_paymt_formula	The Interest Payment Formula table contains information necessary to calculate interest payments on an instrument. The information must exist in this table or the Interest Payment Schedule table. Data for this table is submitted quarterly, if relevant. Reference: RBC Report Instructions.
int_paymt_sched.dat	int_paymt_sched	The Interest Payment Schedule table contains information necessary to calculate interest payments on an instrument. The information must exist in this table or the Interest Payment Formula table. Data for this table is submitted quarterly, if relevant. Reference: RBC Report Instructions.
mbs.dat	mbs	The MBS table contains information for each single class mortgage-backed security not issued by the Enterprise and, at the Enterprise's discretion, any issued by the Enterprise. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
mbs_arm.dat	mbs_arm	The MBS ARM table contains information for each single class mortgage-backed security in the MBS table that is backed by adjustable rate mortgage (ARM) loans or balloon loans with a reset option. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
mrb.dat	mrb	The MRB table contains information for each mortgage-related security not contained in the MBS or MultiClass Derivative tables. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
multi_class_derv.dat	multi_class_derv	The MultiClass Derivative table contains information for all multi-class and other MBS (REMICs and MBS Strips) that are available through the Intex modeling service. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
mf_data_elements.dat	multifamily_data_elements	The Multifamily Data Elements table contains information specific for each multifamily mortgage owned, or underlying securities issued, by an

RBC – Related Data Tables **Install Location = $RBC_INSTALL_ROOT/db/data/Acme(/lookup)**		
File Name	**Data Table Map**	**Description**
		Enterprise. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
option_sched.dat	option_sched	The Option Schedule table contains information for each option embedded in an instrument and each stand-alone option that is not an option on a futures contract. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
ota.dat	ota	The OTA table contains information that represents the Enterprise starting position as of the report date. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
perf_hist.dat	perf_hist	The Performance History table contains information about the net position of all trades for each non-mortgage instrument other than futures and options on futures. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
prin_chng_sched.dat	prin_chng_sched	The Principle Change Schedule table contains information about the scheduled payment of principle for a given instrument other than payment at maturity. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
rbc_acct_recon.dat	rbc_accounting_reconciliation	The RBC Accounting Reconciliation table contains accounting information to ensure total reported assets equal reported liabilities plus equity, and to ensure the amounts contained in the RBC Report database reconcile to the Enterprise General Ledger. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
reference_asset.dat	reference_asset	The Reference Asset table contains information about instruments whose principle or notional amount declines according to the amortization of a mortgage pool or mortgage-backed security. Data for this table is submitted quarterly, if relevant. Reference: RBC Report Instructions.
sf_cmt_data_elements.dat	sf_cmt_data_elements	The Single Family Data Elements – Commitments table contains information specific for each single-family mortgage commitment owned, or underlying securities issued, by an Enterprise. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
sf_data_elements.dat	single_family_data_elements	The Single Family Data Elements table contains information specific for each single-family mortgage owned, or underlying securities issued, by an Enterprise. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
trade_hist.dat	trade_hist	The Trade History table contains information about each transaction of each non-mortgage instrument other than futures and options on futures. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
whole_loan_master.dat	whole_loan_master	The Whole Loan Master table contains information for each owned or guaranteed single-family and multifamily mortgage. Data for this table is submitted quarterly. Reference: RBC Report Instructions.
wl_master_cmt.dat	wl_master_cmt	The Whole Loan Master -Commitments table contains information for each owned or guaranteed single-family and multifamily mortgage commitment. Data for this table is submitted quarterly. Reference: RBC Report Instructions.

Table 2-13: RDM Data File Samples

RDM – Related Data Tables **Install Location = $RBC_HOME/db/data/Acme/lookup**		
File Name	**Data Table Map**	**Description**
income_statement.dat	income_tax_rate	Stores the income tax rate for each year of the stress period for a run profile. File contains needed seed values.
int_rate_cat_type.dat	interest_rate_category_type	Look-Up table for interest rate categories.
mort_prod_types_type.dat	mortgage_product_types_type	Look-Up codes to distinguish different financial instruments.
multi_family_type.dat	multi_family_type	Look-Up table for multi-family financial instruments. Identifies product type interest rate category.
report_profile.dat	report_profile	Stores the report profile name and the associated report specifications for each report profile. File contains needed seed values.
run_profile.dat	run_profile	Stores inputs, runtime parameters, and other details for each run. File contains needed seed values.
single_family_type.dat	single_family_type	Look-Up table for single-family financial instruments. Identifies product type interest rate category.

3. SOFTWARE ORGANIZATION

3.1 Database Components

3.1.1 Database

Reference *The Risked-Based Capital Simulation Application Installation Manual* for the database configuration information.

3.1.2 Data

The RBCSIM database is comprised of objects that support the following four components:

- *Risk-Based Capital (RBC) Report Instructions* (quarterly data submission)
- RBC look-up tables
- Report Decision Module (RDM)
- Data Validation System

The *Risk-Based Capital Report Instructions* specify the mortgage, mortgage-related and non-mortgage information that the Enterprises are required to submit to OFHEO on a quarterly basis.

The RBC look-up tables contain additional information needed to run the stress test model in accordance with the Risk-based Capital Rule. The interest rates and house price index (hpi) values are updated each quarter to reflect the current economic condition. The remaining look-up tables contain static values that are explicitly defined the in the Risk-based Capital Rule.

The RDM data tables are used to support the up-rate/down-rate scenario testing for each Enterprise. The cash flow files generated in a separate step (using Enterprise quarterly submission and the RBC look-up data) are read into the RBCSIM RDM module and the resulting financial statements are written to these data tables. The RDM is run with each quarterly submission.

The Data Validation System performs a variety of checks on the quarterly submission data and writes the results to supporting data tables. These validation checks are required to ensure the data is model ready.

3.2 Application Components

The simulation model calculates the RBC Capital Requirement. The model is composed of three modules that project cash flows and a module that performs accounting, tax, investment, funding, financial reporting and capital calculations. All of the modules are implemented in C++ and are executed from the command line using a set of configuration files. These modules are : The Whole Loan Cash Flow (WLCF) module; The Mortgage Related Securities (MRS) module; The Non-Mortgage Instruments (NMI) module; and the Reporting and Decisions Module (RDM).

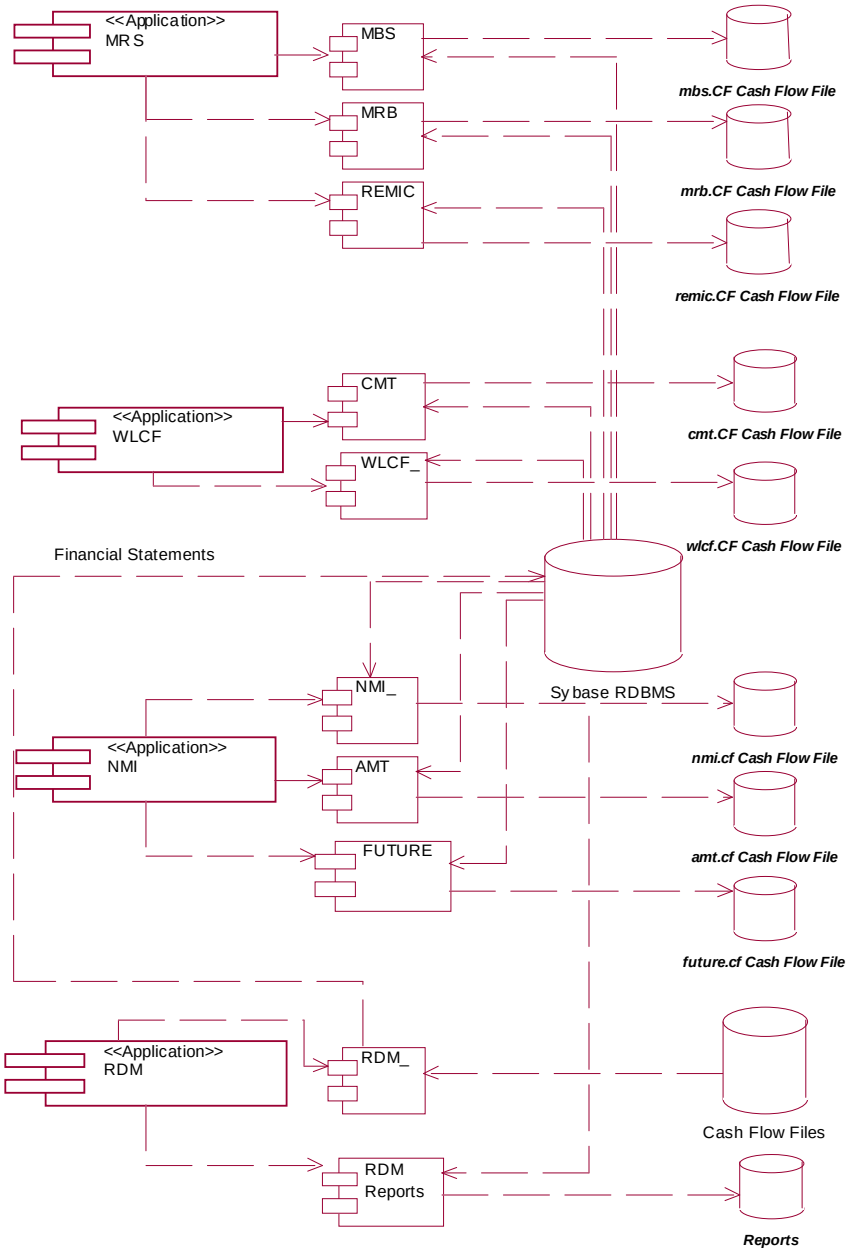


Figure 3-1: Component Diagram

3.2.1 MRS

The Mortgage Related Securities (MRS) module projects cash flows for single class MBS, Mortgage Revenue Bonds (MRBs) and multi-class MBS (e.g. REMICs, Strips, etc.). The MRS module uses the proprietary Intex Solutions, Inc. API to project the cash flows for multi-class MBS. The MRS module reads information from the configuration file and input data from the database, performs amortization, default and prepayment calculations and writes its output to a text file as projected cash flows. Section 3.7 of the RBC Rule provides a detailed discussion of the calculations performed by the MRS module. The MRS module operates in three modes. One mode produces cash flows for single class MBS (MBS), the second mode produces cash flows for MRBs (MRB) and the third mode produces cash flows for multi-class MBS (REMIC).

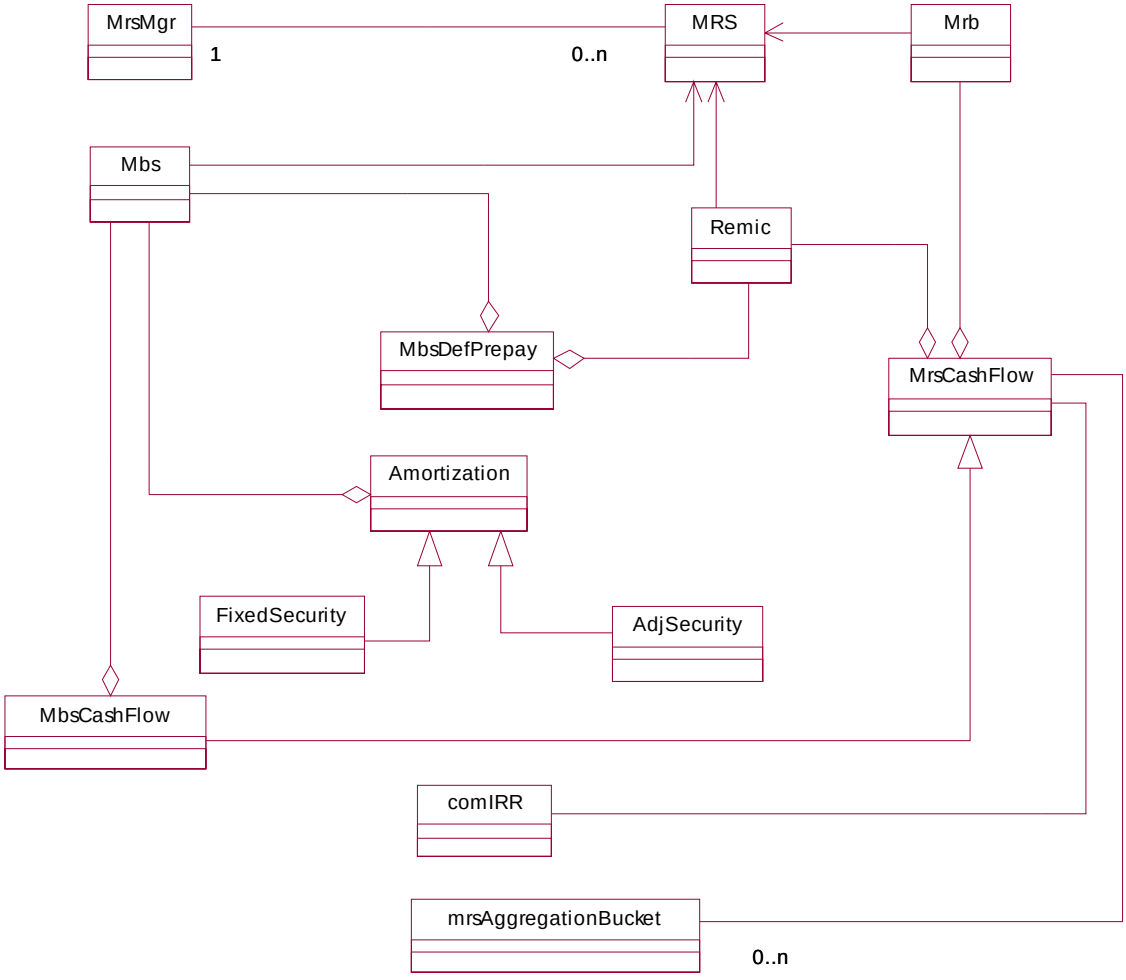


Figure 3-2: MRS Class Diagram

3.2.2 WLCF

The Whole Loan Cash Flow (WLCF) module projects cash flows for retained and sold whole loans as well as for commitments. The WLCF reads information from the configuration file and input data from the database, performs amortization, default and prepayment calculations and writes its output to a text file as projected cash flows. Sections 3.2 and 3.6 of the RBC Rule provide a detailed discussion of the calculations performed by the WLCF module. The WLCF operates in two modes. One mode produces cash flows for retained and sold whole loans (WLCF); the other mode produces cash flows for commitments (CMT).

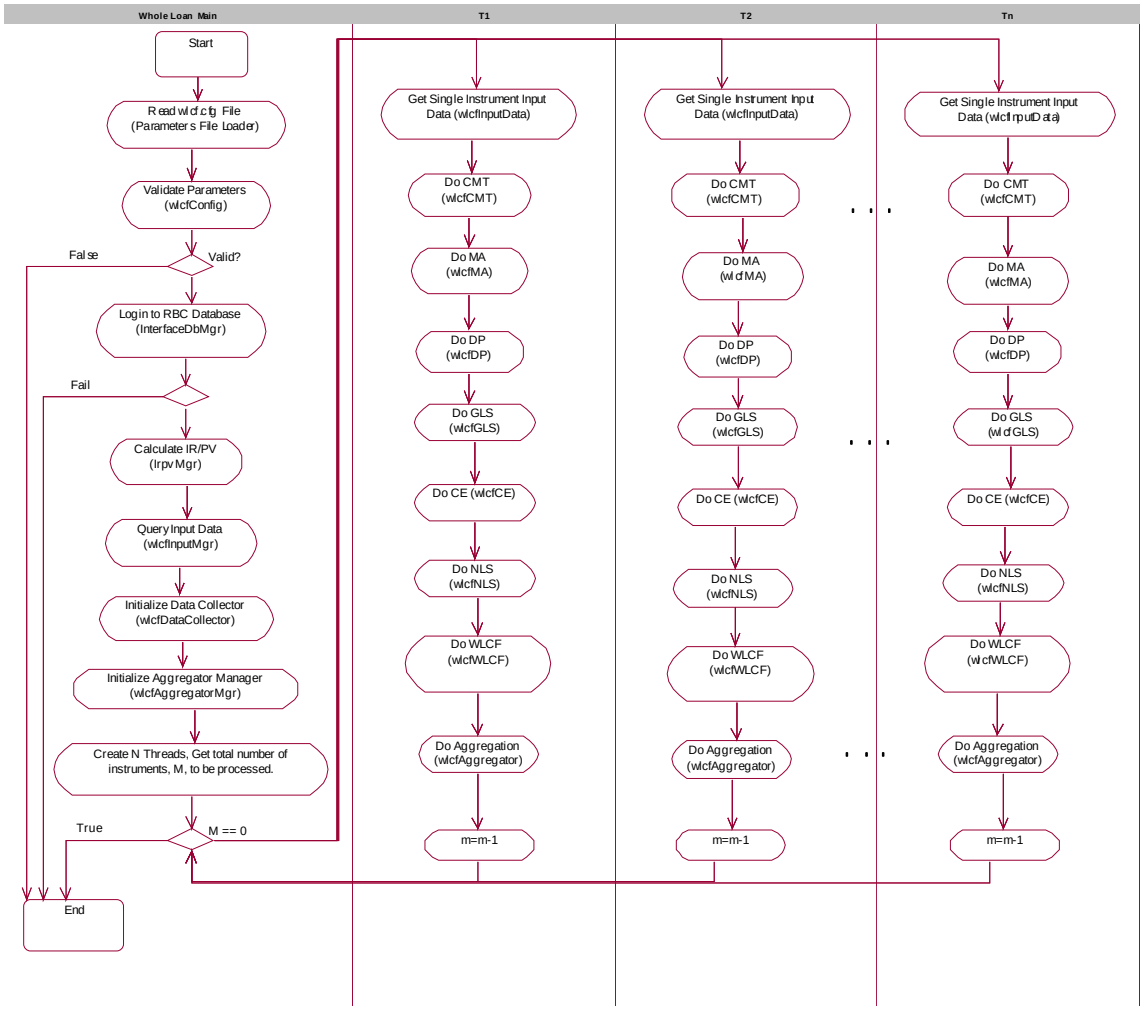


Figure 3-3: Whole Loan Cash Flow Software Flow Diagram

3.2.3 NMI

The Non-Mortgage Instruments (NMI) module projects cash flows for debt instruments, non-mortgage related investments, guaranteed investment contracts, preferred stock, and derivative contracts. The NMI module also performs the Alternative Modeling Treatment (AMT) calculations. The NMI module uses the proprietary Intex Solutions, Inc. API to project the cash flows for non-mortgage-related ABS (e.g. auto loans, credit cards, etc.). The NMI module reads information from the configuration file and input data from the database, performs financial calculations and writes its output to a text file as projected cash flows. Sections 3.8 and 3.9 of the RBC Rule provide a detailed discussion of the calculations performed by the NMI module. The NMI operates in three modes. One mode produces cash flows for Futures (FUT), the second mode produces cash flows for items subject to AMT (AMT), and the third mode produces cash flows for all other instruments (NMI).

3.2.4 Reporting and Decisions Module (RDM)

The RDM performs two main functions:

- **GAAP Accounting and Reporting** – The cash flow files provide the detailed financial instrument cash flows over the simulation period. Because the required cash flow data elements vary by product type, the RDM includes a series of input routines by product type. The starting position data provides opening balances for certain balance sheet accounts, income statement accounts, and variables necessary to perform decision rules for accounts. The RDM converts the simulation period cash flow and starting position information from a cash basis to an accrual basis. The converted cash flows are booked in a journal by debiting and crediting the appropriate general ledger accounts. The journal bookings are combined with financial starting position data and user controlled managerial decision criteria (the second of the two main functions) to produce detailed pro forma financial reports that are reasonably close to GAAP reporting.
- **Funding, Investing, Taxes, and Dividend Payments and Stock Repurchases** - The RDM implements the funding, investing, and stock and dividend payment algorithms defined in the RBC Rule. It also calculates taxes and the capitalization status of each Enterprise.

Figure 3-5 provides a view of how data flows through the RDM processes. Figure 3-6 shows the sequence of method execution that produces financial statements.

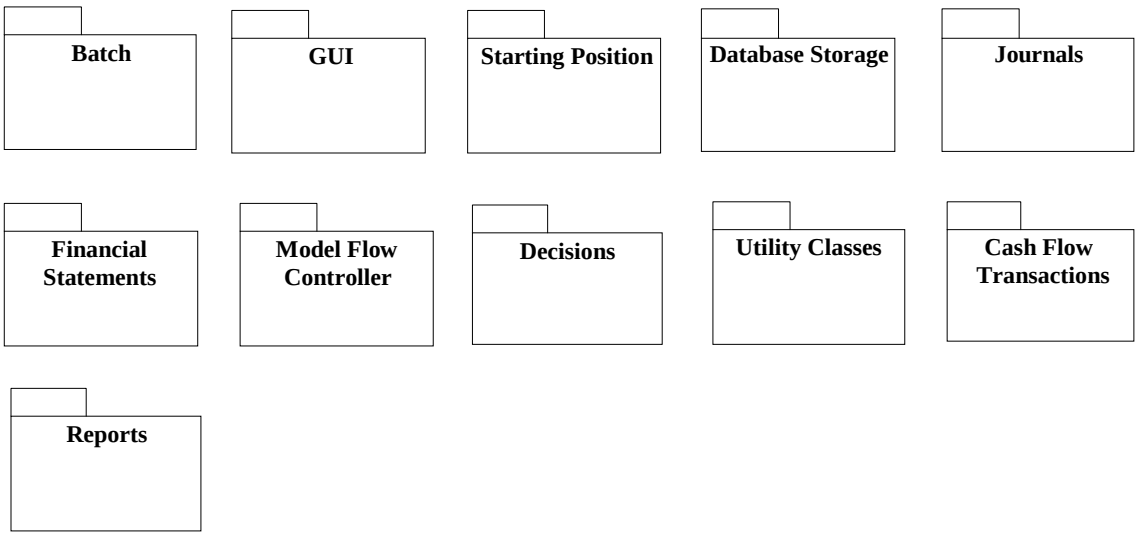


Figure 3-4: The RDM Components

3.2.4.1 Financial Statements

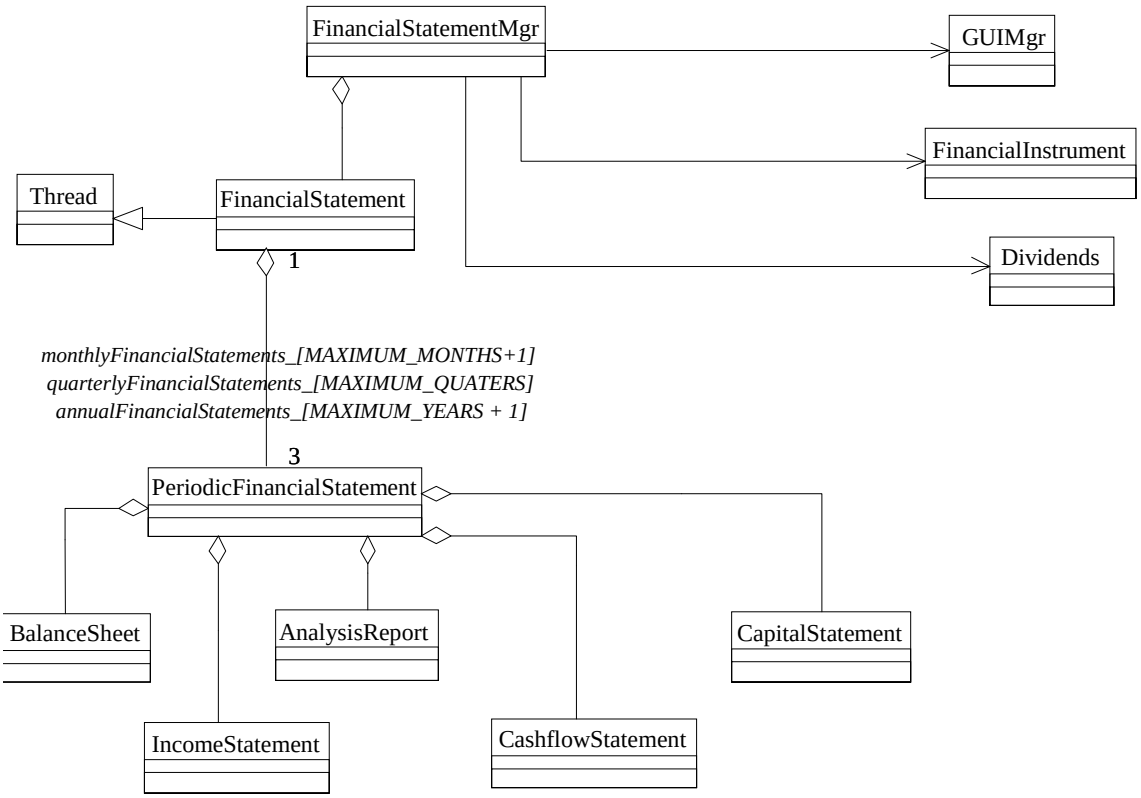


Figure 3-5: Financial Statement Class Diagram

FinancialStatement contains 3 variations of PeriodicFinancialStatements:

1- Annual Financial Statements (11 years, year 0 +10 model years)

2- Quarterly Financial Statements (41 Qtrs, quarter 0 + 40 model Quarters)

3- Monthly Financial Statements (121 months, month 0 + 120 model months)

The Annual and quarterly Financial Statements are calculated from the monthly Financial Statements.

PeriodicFinancialStatement is used to model the Annual Financial Statement, the Quarterly Financial Statement, and the Monthly Financial Statement.

The PeriodicFinancialStatement contains an Income Statement, a Balance Sheet, a Cashflow Statement, an Analytics Report and a Capital Statement.

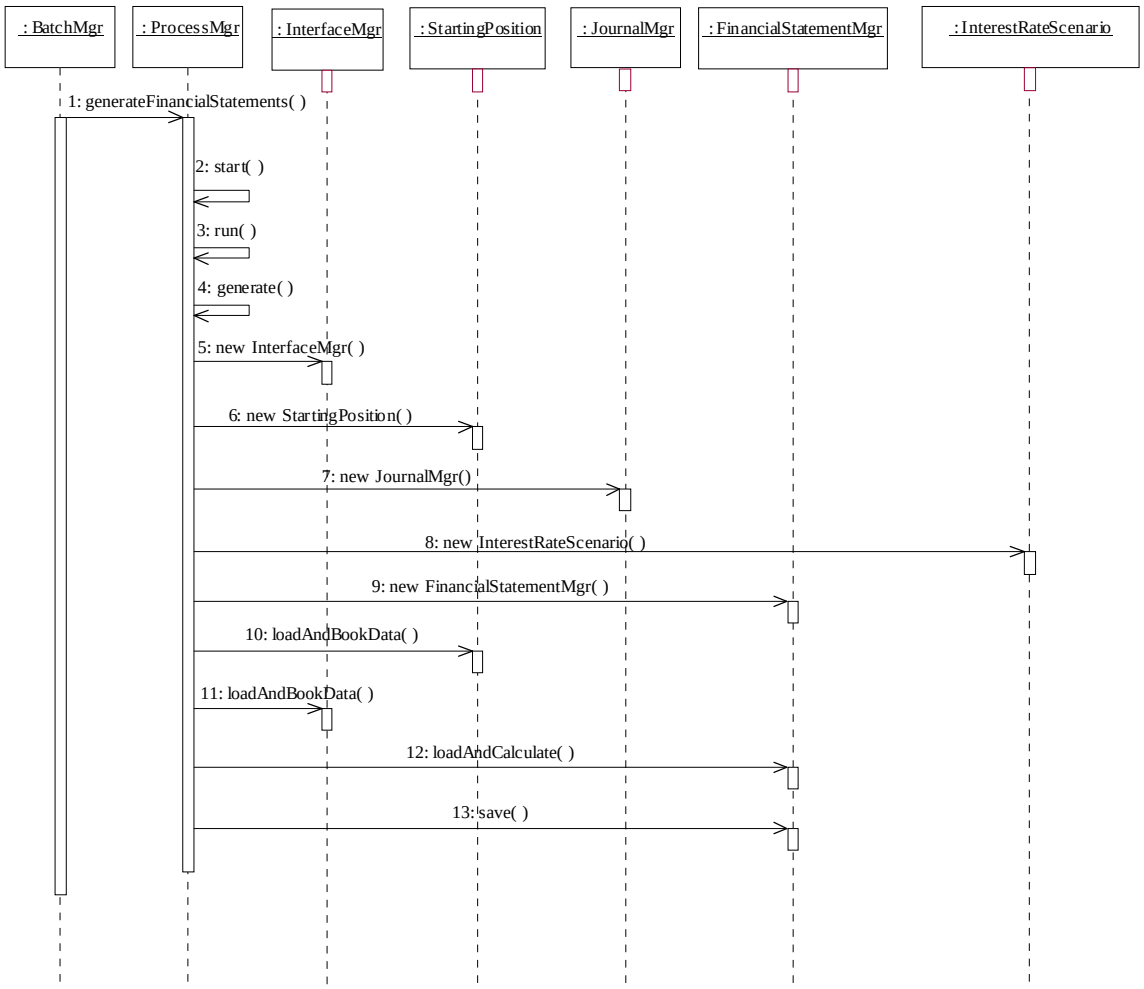


Figure 3-6: Financial Statement Generation Sequence Diagram

Figure 3-6 illustrates the process flow for the creation of the financial statement.

- 2~4: start() is a virtual function inherited from the Thread class. Separate threads are created for generating financial statements and printing reports.
- 5~9: Necessary manager objects are created.
- 10: Data is loaded and booked from the starting position.
- 11-12: CashFlows are booked.
- 13-14: Financial Statements are calculated and saved.

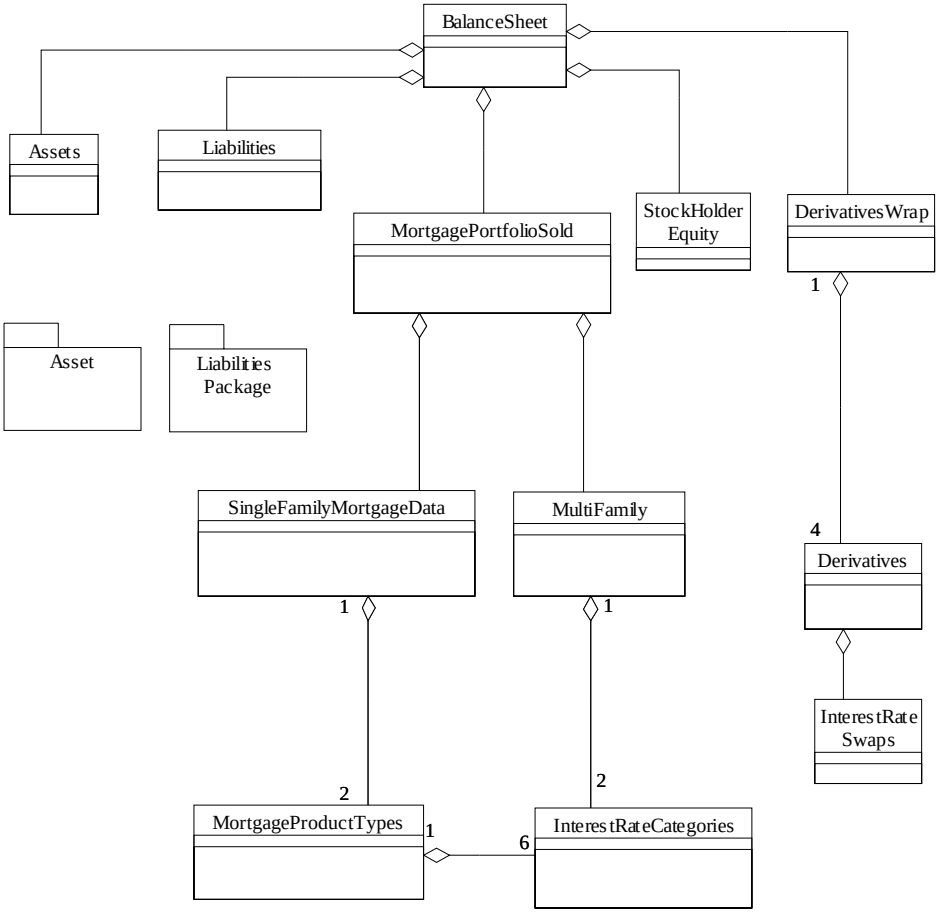


Figure 3-7: Balance Sheet Class Diagram

BalanceSheet class calculates and saves the balance sheet for the Periodic Financial Statement.

The BalanceSheet class has an aggregate relationship with the classes above. An aggregate relationship depicts a whole to part relationship. For example, MortgagePortfolioSold, Liabilities, StockHolderEquity, Assets, and DerivativesWrap classes are part of the BalanceSheet class.

The BalanceSheet class also utilizes classes from the Common Components package.

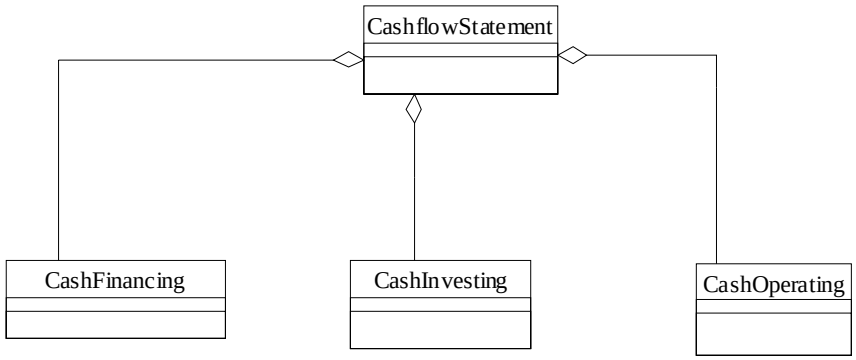


Figure 3-8: Cashflow Statement

The CashflowStatement class has an aggregate relationship with the following classes: CashFinancing, CashInvesting, and CashOperating classes.

CashflowStatement class calculates and saves the cash flow statement for the Periodic Fiancial Statement.

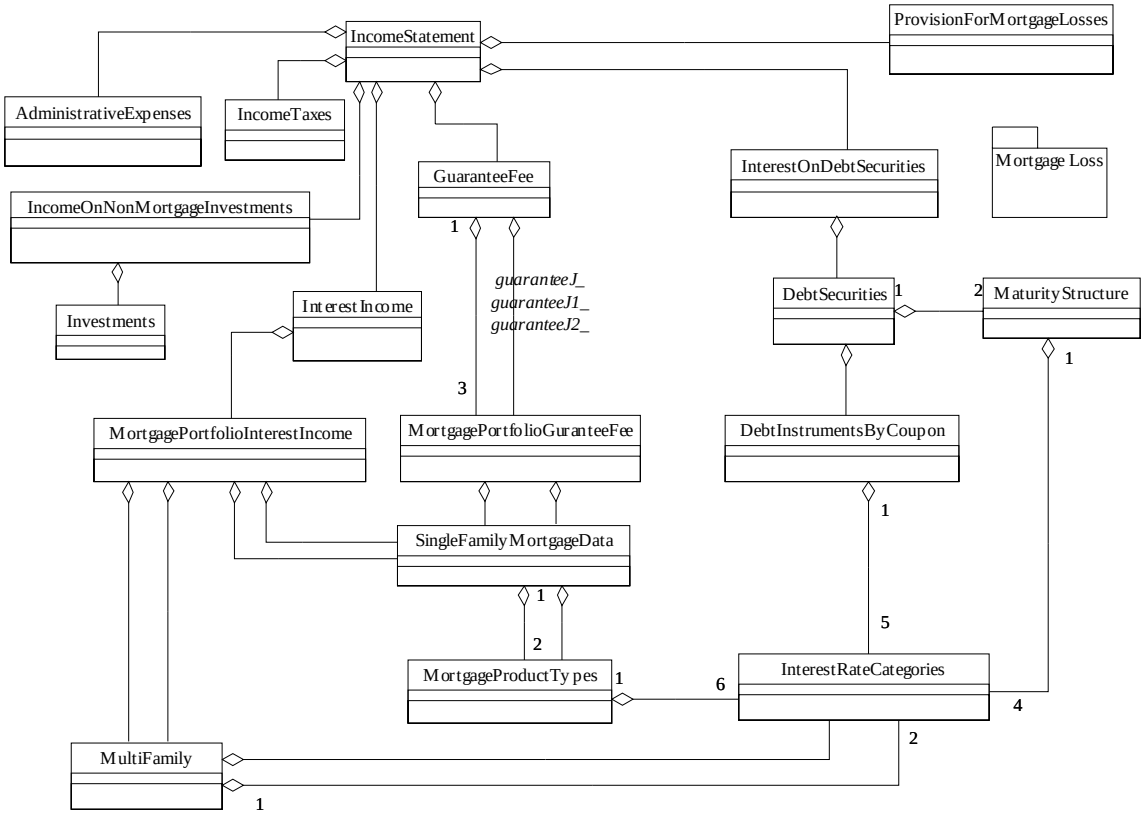


Figure 3-9: Income Statement

The IncomeStatement class calculates and saves the Income Statement for the Periodic Financial Statement.

The following classes have aggregate relationships with the IncomeStatement class: IncomeTaxes, MortgagePortfolioInterestIncome, IncomeOnNonMortgageInvestment, Dividends, ProvisionForMortgageLosses, AdministrativeExpenses, GuaranteeFee, InterestIncome, and InterestOnDebtSecurities. Many of these private members utilize classes from the Common Components package, specifically: InterestOnDebtSecurities, Investments, MortgagePortfolioInterestIncome, MortgagePortfolioGuaranteeFee, and DebtSecurities.

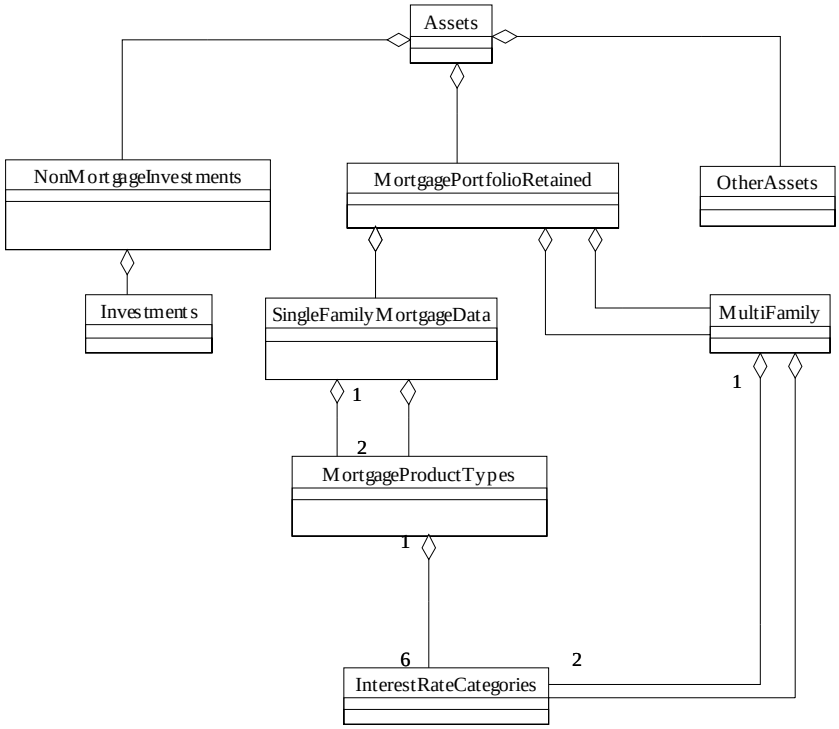


Figure 3-10: Assets Class Diagram

Assets is a container class for the asset information on the Balance Sheet as specified in the Financial Statement Document.

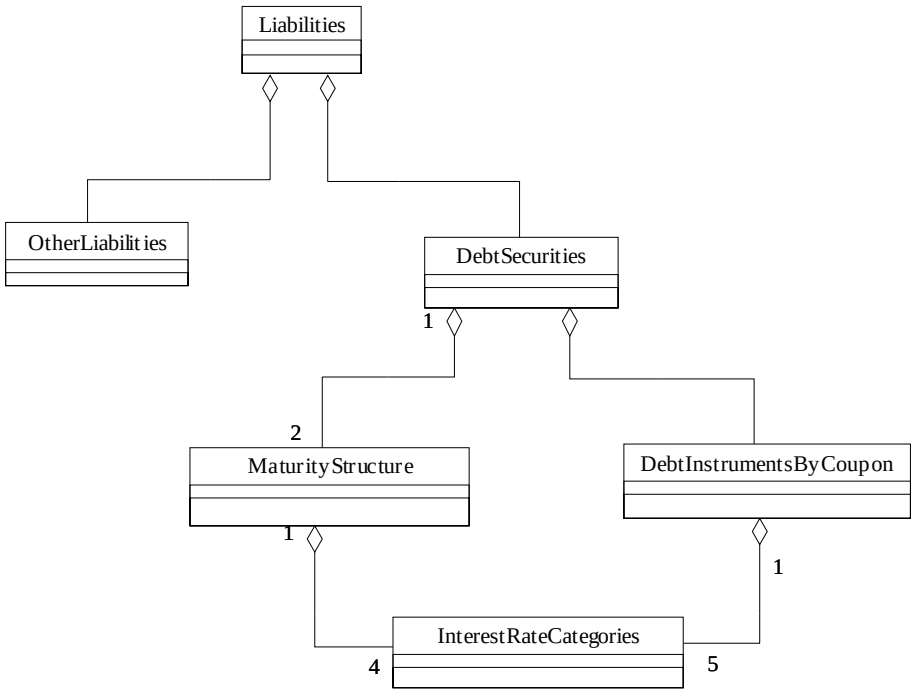


Figure 3-11: Liabilities Class Diagram

Liabilities is a container class for the liability information on the Balance Sheet as specified in the Financial Statement Document.

The Liabilities class has an aggregate relationship with OtherLiabilities, and it utilizes DebtSecurities from the Common Components package. The DebtSecuirties class utilizes MaturityStructure and DebtInstrumentsByCoupon, both of which are from the Common Components package.

4. APPLICATION BUILD AND DEPLOYMENT

The application build and deployment process consists of specific component and sub-component Makefiles, a master Makefile, the Common.mak file, the rbc_build.bash script, the rbc_install.bash script, and the rbc_install.bash script.

The Common.mak file contains installation specific library and include file locations. This file is included by all of the component make files.

4.1 The "rbc_build.bash" Script

The rbc_build.bash script is a compilation and build script which executes the appropriate RBCSIM application component Makefile. In order to execute the build, $RBC_HOME must be set and Common.mak in common/build must be configured (reference section 4.2.1, Step 3 of *The Risk-Based Capital Simulation Application Installation Manual*).

4.1.1 The "rbc_build.bash" Synopsis

rbc_build.bash

[-target *target_name* (default=parallel)]	: Target to Build
[-help]	: displays usage

where target_name is one of the following:

params, common, dbmgr, ir_pv, mrs, nmi, whole_loan, rdm_Common, rdm_FS, rdm_FI, rdm_Journal, rdm_SQR, rdm_decision, rdm_GUI, rdm, clean, all, parallel

Note: If no parameters are entered rbc_build.bash will compile all modules with the parallel option.

4.1.2 The "rbc_build.bash Options

The script file supports the following options:

-help — Prints a usage statement.

-target — The following *Makefiles* / ***Targets*** will be executed when the specified target is provided:

params:	*params/build/Makefile*
common:	*common/build/Makefile*
dbmgr:	*dbmgr/build/Makefile*
ir_pv:	*ir_pv/build/Makefile*
mrs:	*rbc.mrs/build/Makefile*
nmi:	*nmi/build/Makefile*
whole_loan:	*rbc.whole_loan/build/Makefile*
rdm_Common:	*rdm/Common/source/Makefile*
rdm_FS:	*rdm/FS/source/Makefile*
rdm_FI:	*rdm/FI/source/Makefile*
rdm_Journal:	*rdm/Journal/source/Makefile*
rdm_SQR:	*rdm/SQR/source/Makefile*
rdm_Decisions:	*rdm/Decisions/source/Makefile*
rdm_GUI:	Executes the following targets: ***dbmgr, common***; then the Makefile: *rdm/GUI/source/Makefile*
rdm:	Executes the following targets: ***rdm_Common***, ***rdm_FS rdm_FI, rdm_Journal, rdm_SQR, rdm_Decisions, rdm_GUI***.
clean:	Executes the ***clean*** target within all the component Makefiles. This removes all of the .o and binary files generated by the compile.
all:	Executes all of the Component Makefiles sequentially.
parallel:	Performs a dmake (parallel make) on all of the component Makefiles.
Note:	The master Makefile: *rbc.build/bin/Makefile*, is executed first. The *common/build/Common.mak,* is included by the component Makefiles.

4.1.3 The "rbc_build.bash" Examples

- rbc_build.bash
  - Perform parallel build on all targets.
- rbc_build.bash –target clean
  - Cleans all targets
- rbc_build.bash –target whole_loan
  - Perform build on the Whole Loan module

- rbc_build.bash –target rdm
    - Builds all RDM targets

4.2 The "rbc_install.bash" Script.

The rbc_install.bash script is the RBCSIM build and install driver script. This script can build the RBCSIM application by executing the rbc_build.bash script, copy binaries and/or source files to different locations, updates configuration files, and create a compressed file containing the entire RBCSIM content.

4.2.1 The "rbc_install.bash" Synopsis

rbc_install.bash

[-o(ut) *Installation Directory*]	: default: $RBC_INSTALL_TARGET
[-n(ame) *Installation Name*]	: default: *Current Name*, then RBCSIM-V1.0
[-i(n) *Path to input directory*]	: default: $RBC_SRC_HOME
[-src]	: Install source files, default: do not install
[-build]	: compiles source, default: no build
[-code_release]	: prepares code and files for release and compresses files.
[-rmfiles *file_list*]	: remove these files (and file list) from code release delivery
[-help]	: displays usage
[-menu]	: use menu option

Note: If no parameters are entered, the install script will copy the binaries from within the source directories to $RBC_HOME/bin. The $RBC_HOME environment variable must be set in order to perform a no-parameter execution.

4.2.2 The "rbc_install.bash" Options

The script file supports the following options:

-o -out	Allows the user to enter an installation directory path. The default is the $RBC_INSTALL_TARGET environment variable.
-n -name	Allows the user to enter a build or release name. The default is the *Current Name*, the $RBC_HOME directory. If $RBC_HOME is not set, then RBCSIM-V1.0 is the default.
-i -in	Allows the user to enter the RBCSIM application input directory. The default is $RBC_SRC_HOME.
-src	This options enables the source files to be installed. **Note:** The source files cannot be installed to the same directory.
-build	This option compiles source files prior to installing the binaries.
-code_release	This option prepares code and files for release and compresses files.
-help	prints a usage statement.
-menu	This option provides a menu to execute the installation.

4.2.3 The "rbc_install.bash" Examples

- rbc_install.bash
    - copies binary files from the rbc/*component*/bin directories to the RBC_HOME/bin directory.
- rbc_install.bash –build
    - performs: rbc_build.bash –target clean
    - rbc_build.bash *(build all in parallel)*
    - copies binary files from the rbc/*component*/bin directories to the RBC_HOME/bin directory.
- rbc_install.bash –build –o /usr/apps –n RBCSIM-V1.1
    - Performs all of the steps from the previous example.
    - Creates the RBCSIM directory structure at /usr/apps/RBCSIM-V1.1
    - Copies binaries into /usr/apps/RBCSIM-V1.1/bin
    - Updates configuration files and places them into /usr/apps/RBCSIM-V1.1/config.

- *Note: By dotting the rbcenv.bash or sourcing rbcenv.csh file (depending on the users shell), the users can now use the newly installed RBCSIM application.*

5. APPLICATION EXECUTION

The RBCSIM application execution is the run process using the RBCSIM binaries, run-time scenarios, the database, and data sets which have already been loaded and installed. It is assumed that all the steps described in sections 4.2.1, 4.2.2, 4.2.3 and 4.2.4 of *The Risk-Based Capital Simulation Application Installation Manual,* have been executed prior to this step. For details on run-time scenario configuration, reference section 3.2 of *The Risk-Based Capital Simulation Application User Manual.*

5.1 The "rbc_execute.bash" Script

The "rbc_execute.bash" file is a BASH script for running the model.

In the first synopsis form, the script will execute the model binaries using the parameter configuration file provided on the command-line and the "environment.config" file in the location specified by the $FSM_HOME environment variable or in $RBC_HOME/config directory if $FSM_HOME is not defined. The user can specify a different path for the "environment.config" using the '-c' flag. When using this flag, the full pathname, to include "environment.config" must be specified. The configuration file, provided on the command-line, must provide the fill path information for the file also.

The default list of modules is the following:

CMT	-	Commitments
WLCF	-	Whole Loans
MRB	-	Mortgage-Revenue Bonds
MBS	-	Single-Class Mortgage-Backed Securities
NMI	-	Non-Mortgage Instruments
RDM	-	Report, Decision Model

The user can specify which modules to run using the '-r' flag. The list of additional modules that can be run is the following:

REMIC	-	Multi-Class Mortgage-Backed Securities
AMT	-	Alternative Modeling Techniques
FUT	-	Futures

***Note**: data for the above three modules are not provided with the stylized data set.*

When providing a list of modules to run, they should be pipe, '|', delimited with no spaces. For example:

rbc-execute.bash –r WLCF|MRB|FUT|RDM …

will only run the Whole Loan, Mortgage-Revenue Bonds, Futures and RDM modules. The user can also enter a value of "ALL" when specifying the module list. This option will run all the modules.

The script will run the stylized up-rate scenario if a Run-Set is not provided on the command line. Finally, the script assumes the user's database login and UNIX login are the same. The '-u' switch can be used to specify the database login name when the user's database and UNIX login names are different.

The second synopsis form will print the script's usage statement to the console.

The third synopsis form will print a list of available Run-Sets to the console.

5.1.1 The "rbc_execute.bash" Synopsis

rbc_execute.bash [-r *modules*] [-c *environment-config*] [-u *username*] config-file [runset]

rbc_execute.bash –h

rbc_execute.bash -l

5.1.2 The "rbc_execute.bash" Options

The script file supports the following options.

-r	allows the user to specify the list of modules to execute. Valid flag values are: ALL, which runs all modules; ACME, which runs the CMT, WLCF, MRB, MBS, NMI and RDM modules; or the user-provided list of modules. When specifying the list of modules to run, separate them using the pipe ('|') character with no spaces.
-c	allows the user to specify the "environment.config" file to be used for the run. The full path must be provided to include the filename, which must be "environment.config".
-u	allows the user to specify the database login name to use for the run.
-l	prints a listing of available Run-Sets that can be executed.
-h	prints a usage statement.

6. OUTPUT RESULTS

6.1 Cashflow Files

For details on the Cashflow files, reference Section 4.1 of *The Risk-Based Capital Simulation Application User Manual.*

6.2 Logs

For details on the Log files, reference Section 4.4 of *The Risk-Based Capital Simulation Application User Manual*.

6.3 Database Updates

The following tables will be populated with the financial statements and capital requirement reports generated by the RDM:

- administrative_expenses
- assets
- balance_sheet
- capital_statement
- capital_statement_static
- cashflow_financing
- cashflow_investing
- cashflow_operating
- cashflow_statement
- checked_reports
- credit_and_loan
- debt_instruments
- debt_securities
- derivatives
- dividends
- earning_ratios
- financial_statement
- income_on_non_mortgage_invest

- income_statement
- income_tax_rate
- income_taxes
- interest_on_debt_securities
- interest_rate_swaps
- investments
- liabilities
- loss_portfolio
- maturity_structure
- mortgage_portfolio_gurante_fee
- mortgage_portfolio_int_income
- mortgate_portfolio_retained
- mortgage_portfolio_sold
- mortgage_product_types
- multi_family
- non_mortgage_investments
- other_assets
- other_liabilities
- other_ratios
- periodic_financial_statement
- provision_for_mortgage_losses
- run_profile
- single_family
- stock_holder_equity
- yields_and_costs