Lehman Brothers: Risk Reporting and Technology

A Major Qualifying Project

(Project Code: MXC-W052)

Submitted to the faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

William Hays

Amy Jackson

Jason Tondreau

Igor Ushakov

Approved by:

Professor Arthur Gerstenfeld, Department of Management

Professor Michael Ciaraldi, Department of Computer Science

Professor Jon P. Abraham, Department of Actuarial Mathematics

December 13, 2005

Preface

This report is a combination of two Lehman Brothers projects – one project regarding country risk reporting and the other consolidating foreign exchange spot credit limits. The two projects are related in some ways and unrelated in others. The introduction and background are two common sections. All other sections of this report are project-specific.

The project-specific sections are divided into "Part 1" and "Part 2." Part 1 relates to the country risk reporting project done in New York, whereas Part 2 relates to the project completed in London, regarding foreign exchange spot credit limit consolidation.

Abstract

The main goal of this project was to improve risk reporting and technology at Lehman Brothers for both country and credit risk. The project was divided into two separate parts that were completed in parallel. One part was completed in New York which focused on the improvement of the country risk reporting process. The other was executed in London to consolidate foreign-exchange spot credit limits.

Abstract Part 1

The main goal of this project was to improve on the current system for reporting country risk at Lehman Brothers in New York. The project was separated into two parts. The first was to evaluate the quality of the data going into the reports by running a series of analyses to test the accuracy of this information. The second part of the project was to automate the country risk reporting process, which will significantly reduce the time it takes to generate the country risk reports. To assist in the automation process we developed a prototype to model what features and capabilities the new system will have.

Abstract Part 2

The project refined the process of comparing foreign-exchange spot credit limits across the Reuters and EBS broker dealer systems and the Lehman Brothers international internal Credit Work Station system. We developed a system to store and access all needed information quickly. Utilizing this new system, we developed a new set of improved process steps to replace the prior steps. These new steps were more efficient and more automated. The implemented system enabled foreign-exchange spot credit limit information to be seen globally in one consolidated view. Upon completion of the project, the system was integrated into a common credit risk management portal and was used in a production environment by Lehman Brothers' credit risk management department.

Acknowledgements

Our group would like to thank the following individuals and groups for their time, effort, support, and guidance in completing this project:

From Lehman Brothers:

Mahvish Ayoob
Scott Burton
Scott Chang
Daniel Eisen
Andrew Fox
Jocelyn Girard
Felicia Goldring
Linda Landy
Thomas McConnon
Anup Menon
Christian Moeller
Brendan Murphy
Robert Nines
Jesus Rego
Prasad Sabbineni
Ford Tan
Simon Wilkins
Leesan Wong
Charis Yang

From Worcester Polytechnic Institute:

Jon P. Abraham
Michael Ciaraldi
Arthur Gerstenfeld

Authorship Page

The project began as a collaborative project focusing on one goal and split early to become two separate projects. The researched background information was done as a whole, while the separate projects were done by their respective groups. Editing efforts were made by all to compile this document into one polished work.

The sections of this document separated by "Part 1" or "Part 2" correspond to sections of the report for the New York based project and the other for the London based project, respectively. Part 1 was completed by Jason Tondreau and Igor Ushakov. Part 2 was done by William Hays and Amy Jackson. Sections not separated by a part relate to both projects.

Executive Summary Part 1

Many financial institutions, such as Lehman Brothers Incorporated, are continually emphasizing the development of an improved means for reporting financial risk. Ideally, Lehman Brothers will be able to view accurate, real-time, risk information, on a daily basis. Lehman Brothers is striving to produce this information to support high level financial risk decision making. However, the existing process for reporting country risk at Lehman Brothers is labor intensive and time consuming. In an effort to alleviate some of these issues Lehman Brothers is in the early stages of developing an enhanced country risk report process. To help achieve this, the primary focus was to assist in the initial phases of the project by analyzing and evaluating existing data sources used to create the reports. This was done in conjunction with the development of a preliminary prototype of the user-interface that would display the reports to the end users.

Prior to working on the improvement of the country risk reporting process, we had to establish an understanding of the current process and the inefficiencies associated with it. The basic research involved reading over existing documentation outlining the steps for creating the reports. To support this, several interviews were conducted with employees that are involved in the development of the country risk reports. Next, we began working closely with several of the employees working on the development of the new process. The project is divided into two fundamental parts that address some of the major issues concerning current country risk reporting. First, the project analyzes the data being used for the reports. Second, it develops the way that the new reports will be automated and displayed to the end users. In looking at the data, there are several plausible options to gather accurate, timely information. Initial research has discovered that other internal Lehman systems are already producing a majority of the data that could potentially be used in the new country risk reports. The success of the new reports relies heavily on the information it is comprised of. Due to this, a significant part of the project was helping to validate data from other sources Lehman already has access to. This data was compared directly to the information needed to compile a complete and accurate country risk report. In parallel, the prototype of the user interface was being developed not only to get feedback from the users, but to also serve as a starting point in the future development of this project.

Preliminary research done both during and prior to this project has shown that much of the data needed for the country risk report is contained in existing Lehman databases. Furthermore, the data generated daily is accurate which requires far fewer manual adjustments. However, some data research has identified several anomalies and/or gaps in information. The development of the prototype was done to give the users a first look at what the finished product could be capable of.

The design of the prototype was done to act as a catalyst for the development of the final program with intentions of getting the users to think about the new systems implementation, while triggering some feedback for potential improvements.

The current quality of information and the rate in which it is received has attracted growing concern in the country risk department at Lehman Brothers. Initial research and experimentation have shown promising results in obtaining prompt and accurate information. However, since the overall project is still in its early phases we recommend that Lehman continues to consider alternative data sources with respect to reliability, validation, and timeliness. Furthermore, proceeding with the development of the user interface will continue to stimulate new ideas for enhancements while fostering innovation in the development process.

We have also developed a set of recommendations on behalf of Lehman Brothers for the future progress of the country risk reporting project. In order to improve the quality of the data going into the country risk reports we have developed the following recommendations.

- Decide what methodologies need to be applied to each product type
- Apply different methodologies for each product
- Incorporate product type data from SUMMIT and HJM to Country Risk FX
- Incorporate all countries, not only emerging market countries, into Country Risk
- Include the data dictionary in the new country risk reporting program
- Use this same data analysis system for any other types of products with similar misclassification issues

We have also outlined several recommendations to help in the future of the development of the new country risk reporting system.

- Develop a Project Plan and tune it over time
- Continue the process in the iterative manner
- Open communication with users on each step
- Reorganize the database structure

Incorporating these recommendations on both a short and long term basis will help to improve the process in with the new system is developed.

Executive Summary Part 2

One of the product lines the Lehman Brothers credit reporting team is concerned with is foreign-exchange (FX) spot credit limits. There are two different trading applications (Reuters and Electronic Broking Service) where these FX credit limits are maintained for spot matching products. Prior to this project, the process for comparing these applications' FX spot credit limits with Lehman's internal credit analysis and reporting system (Credit Work Station) is time consuming and requires multiple steps. Also, a system did not exist that verifies that the sum of the FX spot credit limits stored in EBS and Reuters systems is within the limit set and recorded in CWS by a Lehman Brothers analyst. The main objective of this project was to improve this FX credit limit comparison process and create a system that shows the consolidation of foreign-exchange spot credit limits across different broker service systems. To complete this main objective, three goals were set:

1. Formulate a system that allows the **total FX spot credit limit allocation** for each client to be viewed.
2. Create a **logging process** within this system to keep track of each client's investigation status.
3. Develop a way to assure that the **sub-entities** of an entity match up between the two trading applications, Reuters and EBS.

Along side these goals, three "sub-goals" were set:

1. Build an **interface** for the new system as a subsection of the CRM webpage.
2. Have an **internal technology member** that can maintain and further update the new system.
3. Obtain **knowledge** about FX trading, credit limits, and processes purpose.

The process for reaching these goals consisted of several steps. We first had to gain an understanding of the current process, its sequence of steps, and the systems involved. During the next phase, we designed the new process, the database structure, and the system's interface. From these designs, we implemented the new system and its web interface, which was integrated into the Lehman Brother's Credit Risk Management website. Upon completion of implementation, the new system was tested, staged in Q &A, and then moved into production. Along with this system, we left Lehman Brother's with finalized diagrams and procedure documentation.

After completing this methodology, we produced various results. Through understanding the current process, we produced documentation describing the users' needs and steps. We then developed the design specifications for the new system. These design specifications include

functional requirements of the system and an Entity Relationship diagram showing the data organization. With the design of the system we produced the following deliverables:

- Database structure that stores all the Reuters, EBS, and CWS FX spot credit limit data.
- Perl scripts which import the Reuters and EBS FX credit limit data.
- Interface within the Lehman Brothers CRM website implemented in Java Server Pages.
- Documentations for all aspects of the implemented system.

To aid with the understanding, maintaining and use of these deliverables we created diagrams, procedure documentation, and notes for the technology department.

In conclusion, the project goals were met and the new system produces a consolidated global view of the FX spot credit limits from three different data sources; Reuters, EBS, and CWS. The new system has many advantages and functionality. The new system:

- Maps efficiently the clients from the broker systems to CWS.
- Has the ability to log all the client investigation history and show the comparison of Reuters and EBS sub-entities.
- Is conveniently integrated into the Lehman credit risk framework (CRM website).
- Can be found in a common credit risk management portal that allows access control, providing security.
- Was handed over to Bappa Roy, a member of the Lehman Brothers' credit risk technology team.

Through accomplishing these goals we also developed an understanding of Lehman's software development cycle, FX trading and credit limits. Overall, the system makes the process of comparing FX spot credit limits more automated and reduces work repetition.

Based on our results, we provided Lehman Brothers with various recommendations. Our recommendations are broken down into two categories: post project and long term. We recommend the following post project recommendations are completed as soon as possible:

- ***Periodically updated feeds from Reuters and EBS***

  To assure the database is up to date, it needs to contain current FX spot credit limit data. We recommend that this data is requested and received from Reuters and EBS on a regular basis and the database is continuously updated.

- ***New system maintenance***

  To clean-up the new system's information we suggest all the unlinked entities from Reuters and EBS are linked to their matching CWS counterparty name. Along with this,

counterparties with a total FX spot credit limit exceeding the CWS set limit should be investigated. Last, we suggest the history action codes, used in the counterparty history feature, are standardized. All of these recommendations assist in the maintenance of the new system.

Once the project was completed, we were able to make the following long term recommendations, which are more generalized and of less urgency:

- ***Automated feeds from Reuters and EBS***

  Currently, the current FX credit limit data feeds from Reuter and EBS are received via email upon request. We recommend Lehman Brothers sets up with Reuters and EBS automated feeds sent to them on a consistent basis. This will guarantee the database's information is current.

- ***Readily available CRM implementation standards***

  Implementation standards for integrating into the CRM website were not readily available at the execution of this project. We suggest the architecture and framework for the CRM website is more easily accessible so these architecture details will be used throughout, by all implementers.

- ***Global use of the new system***

  Presently, the new system is being used in London. Seeing the new system also contains the FX spot credit limits maintained in New York and Tokyo, we recommend the system is used globally.

- ***New system extension***

  As of now, the new system contains FX spot credit data from Reuters and EBS. This system can be extended to include other product lines and broker dealer systems. For example, the new system can be easily modified to include FX forward credit limits.

- ***WPI project continuation***

  Based on the success of this project and its results, we suggest future WPI projects are taken on by Lehman Brothers for the benefit of credit reporting in London.

If Lehman Brothers follows our previous listed recommendations, FX credit limits will be globally maintained. This will assure the FX credit limits set are accurate and easily accessible. Also, it will aid the credit reporting system, overall improving Lehman Brothers foreign-exchange trading.

Table of Contents

List of Figures

1 Introduction

Many financial institutions, such as Lehman Brothers Incorporated, are continually emphasizing the development of improved means for reporting risk analysis. Country risk is the possibility of financial loss associated with a foreign investment. There are a large number of factors that can influence the amount of risk in any given country. High-quality country risk reports often include in-depth examinations of a country's political, financial and economic uncertainty. A second aspect of risk, also in focus in this report, is credit risk. Credit risk is the possibility of financial loss due to a counterparty defaulting.

Lehman Brothers, an industry leader in global finance, is becoming increasingly interested in country risk reports. Lehman Brothers serves the financial needs of governments, cities, companies, institutional clients, and high net worth individuals all over the world. This responsibility drives the Lehman Brothers Corporation and similar economic organizations to continue to improve their research processes. Developing daily accurate Country risk reports is one of Lehman Brothers' concerns.

Another facet of Lehman Brothers is its credit risk division. Managing the credit risk within Lehman Brothers enables them to execute trades with many counterparties safely, taking into account that counterparty's risk. Within credit risk, there is foreign exchange credit risk. Keeping track of these risks is an important aspect of an investment bank.

Lehman Brothers has previously invested in the improvement of the country risk reporting process. Preceding WPI projects were done to analyze the current global risk reporting procedure for market and credit risk. This was completed in conjunction with the use of the program Business Objects, where existing reports were updated and new reporting methods were developed. These reports could be then accessed through a web interface. The project team also provided training in the use of Business Objects and written training guides to sustain future use of the program.

In order to build upon prior work done on risk, Lehman Brothers has sponsored this WPI project to analyze existing country risk feeds and databases. In conjunction with the country risk aspect of this project, a project focusing on the consolidation of foreign exchange credit limits has been sponsored. They want to assure an intuitive and easy-to-use representation of the information desired. These reports are to be generated daily and have the ability to be customized on-demand. We also worked on making these statements more detailed, produced automatically, while incorporating advanced methodologies.

This project was divided into two separate projects, working in parallel. One project was focusing on country risk; the other was focusing on foreign exchange credit limits. The country risk

project was based in New York while the foreign exchange credit limit project was based primarily in London. This was done to take advantage of Lehman Brothers employees at each location.

The foreign exchange credit limit consolidation project had a set list of goals when it was started. The project goals that were defined were to:

- Formulate a system that allows the **total FX spot credit limit allocation** for each client to be viewed.
- Create a **logging process** within this system to keep track of each client's investigation status.
- Develop a way to assure that the **sub-entities** of an entity match up between the two trading applications, Reuters and EBS.
- Build an **interface** for the new system as a subsection of the CRM webpage.
- Have an **internal technology member** that can maintain and further update the new system.
- Obtain **knowledge** about FX trading, credit limits, and processes purpose.

Given our objectives the goal was to assist Lehman Brothers in the creation and implementation of these reports. We accomplished this through a variety of methods. First, we analyzed the current process for which these reports are produced while looking for inefficiencies and problems throughout the procedure. We conducted a series of interviews in support of our observations and to further our understanding of the process in which these reports were produced. The end result was to provide Lehman Brothers with a more efficient means of delivering these reports on a daily basis to help support the financial decision making process. A similar process was followed for the foreign exchange credit limits project. The end result for the credit limits project was a consolidated reporting view for foreign exchange credit limits.

2 Background

2.1 Risk

Risk is defined as the probability of loss (Merriam-Webster, 2005). In order for risk to be present, there needs to be uncertainty and exposure. When measuring risk, there is a variable, such as a portfolio, that is defined. Then the variable's exposure, in the case of a portfolio, is the uncertainty of its market value. This uncertainty comes from the effects of financial factors on this variable. Risk is assessed with the use of risk measurements. Risk measurements are procedures done to calculate risk metrics. The first metric is sigma or volatility, which measures risk by the standard deviation of the probability function of the unexpected outcomes. With this volatility and the covariance between the portfolio return and the market return, beta or systematic risk can be calculated using the formula:

$$\frac{\mathrm{cov}(Z_p, Z_m)}{\sigma_m^2}$$

Also the metric Value at Risk (VAR) measures the effect of volatility and exposure to financial risks. Other metrics include delta and gamma, which measure the first and second derivatives of the exposure to movements in the value of the underlying asset in a derivatives portfolio (Jorion, 2000). In doing this project we are concerned with financial or investment risk; specifically, credit and country risk.

2.2 Credit Risk

Credit risk is the risk of financial loss due to counterparty failure to perform their obligations (Jorion, 2000). Credit limits are one way credit risk is managed. Credit limits are the maximum exposure a firm is willing to risk on a client. Credit limits are set based on the type of trading and the product traded. They are also set based on the creditworthiness, credit ratings, and credit appetite of a client.

In part two of this project, we dealt with foreign exchange (FX) credit limits. Specifically, we looked at the foreign exchange spot and forward product lines. Analysts setting these credit limits, set them based them on foreign exchange trading.

When setting credit limits, analysts review the client's financial reports to assess the creditworthiness of a company. This is determined by considering many factors. One factor is the result of a client's financial analysis. A ratio analysis of the financial accounts is performed. Some examples of the financial ratios are: net worth ratios, debt to equity ratios, and liquidity ratios. Other factors that are considered are the location of the company and its industry sector. Also, specifically

for foreign exchange trading, the fact the company is a Continuous Linked Settlement (CLS) member is considered. CLS allows foreign exchange transactions to be settled within the same day, eliminating settlement risk and leaving only price risk as the risk for the trade. CLS is a "clearing house" for trades and currently there is a list of fifteen eligible currencies.

After an analyst assesses the creditworthiness of a company, they look at the credit rating and credit appetite of the company. Credit ratings are set by external rating agencies as well as Lehman Brothers' own internal system. Lehman Brothers uses the credit ratings set by Standard and Poors and Moody's. As seen below, for the Standard and Poors scale, the highest rating with the lowest risk is AAA and the lowest rating is D, in default.

Investment Grade	
AAA	the best quality companies, reliable and stable
AA	quality companies, a bit higher risk than AAA
A	economic situation can affect finance
BBB	medium class companies, which are satisfactory at the moment

Non-Investment Grade (also known as junk bonds)	
BB	more prone to changes in the economy
B	financial situation varies noticeably
CCC	currently vulnerable and dependent on favorable economic conditions to meet its commitments
CC	highly vulnerable, very speculative bonds
C	highly vulnerable, perhaps in bankruptcy or in arrears but still continuing to pay out on obligations
CI	past due on interest
R	under regulatory supervision due to its financial situation
SD	has selectively defaulted on some obligations
D	has defaulted on obligations and S&P believes that it will generally default on most or all obligations
NR	not rated

Standard and Poors credit rating scale (wikipedia.com)

The table below shows the credit rating scale for Moody's. Where Aaa is the highest rating with the lowest risk and D, in default is the lowest rating.

Investment Grade	
Aaa	Obligations rated Aaa are judged to be of the highest quality, with minimal credit risk.
Aa1, Aa2, Aa3	Obligations rated Aa are judged to be of high quality and are subject to very low credit risk.
A1, A2, A3	Obligations rated A are considered upper-medium grade and are subject to low credit risk.
Baa1, Baa2, Baa3	Obligations rated Baa are subject to moderate credit risk. They are considered medium-grade and as such may possess certain speculative characteristics.

Speculative Grade	
Ba1, Ba2, Ba3	Obligations rated Ba are judged to have speculative elements and are subject to substantial credit risk.
Ba1, Ba2, Ba3	Obligations rated B are considered speculative and are subject to high credit risk.
Caa1, Caa2, Caa3	Obligations rated Caa are judged to be of poor standing and are subject to very high credit risk.
Ca	Obligations rated Ca are highly speculative and are likely in, or very near, default, with some prospect of recovery of principal and interest.
C	Obligations rated C are the lowest rated class of bonds and are typically in default, with little prospect for recovery of principal or interest.

Special	
D	(in default), WR (withdrawn rating), NR (not rated), (P) (Provisional)

Moody's credit rating scale (wikipedia.com)

The ratings show above for both agencies are based on a company's credit and financial information. Lehman Brothers internal credit rating system is a seventeen point scale similar to the Standard and Poors scale that also incorporates the company's internal viewpoint on risk. Specifically, the Lehman Brothers scale is based on quantitative and qualitative analysis of the company. Some factors that help Lehman Brothers in determining a rating are a company's industry, management, capital, earnings, asset quality, and liquidity.

Last, the credit appetite of a counterparty is considered. The credit appetite of a counterparty is the total possible amount that can be traded between Lehman Brothers and that counterparty at any given time.

The analyst looks at the creditworthiness, credit rating and credit appetite of a counterparty and derives a credit limit. Also, a client's documentation is taken into consideration when deriving this credit limit. Client documentation that is considered includes netting and collateral agreements. Netting agreements are agreements of cash flows or obligations. They are contracts settled with net payments. Collateral agreements are when assets are provided to secure an obligation. In FX trading there are bilateral agreements where assets are provided to secure a two-sided obligation. These agreements are enforced to reduce credit exposure to counterparties. Credit exposure describes how large the amount of outstanding obligations a client will have if it defaults.

2.3 Foreign-Exchange Risk

Lehman Brothers is a client driven investment firm. Therefore, Lehman Brothers performs a foreign-exchange (FX) trade to satisfy a client, to facilitate customer trading. Foreign-exchange trading involves different country currencies being traded, enabling international transactions to take place. There are many risks involved in FX trading.

One of the risks that occur during FX trading is settlement risk. Settlement risk is a form of credit risk that happens at the settlement of a transaction. It is the probability that a counterparty will default before going through with the transaction. In FX trading, it occurs because of the time lapse between when exchange transactions are made (money goes out but doesn't come in). Settlement risk came about after the failure of the German bank, Herstatt Bank. On June 26, 1974 the bank was closed down and failed to follow through with some US payments. This is why sometimes settlement risk is referred to as Herstatt risk. Settlement limits are set to account for this risk. A large set limit signifies good visibility or ability to estimate the bank's probability to default.

In FX trading, when forwards are traded there is a forward risk. There are two factors of risk in forward trades. The first is the settlement risk that takes place when the trade is made in the future. The second is the risk that the market moves (market risk calculated with counterparty exposure) and the counterparty defaults. Forward limits are set based on both of the factors of forward risk.

Also, when FX spot trading occurs there are spot limits set. Spot limits are equal to the settlement limits if there are no forwards traded on the time of settlement. If there are forward contracts ending on the same day as a spot trade, then that is factored in and the settlement risk is higher.

Both forward and spot limits are notional limits, which mean they are based on notional amounts. The plus side of notional limits is they are easily understandable; the down side is volatility, or currency fluctuations.

Reuters and Electronic Broking Service (EBS) are two broker service systems that maintain FX credit limits. Reuters is a global system maintaining both FX spot and forward credit limits. EBS is a local system maintaining only FX spot credit limits for a single location. There are separate limits maintained in EBS for Lehman Brothers offices in London, Tokyo and New York. Another system that plays a role in Lehman Brothers foreign exchange trading is Credit Work Station (CWS). CWS is Lehman Brothers' internal credit analysis and reporting interface with a database backend. Contained in CWS is the total credit limits and credit information up to the previous date. CWS contains FX spot and forward credit limits.

Part two of this project involved both foreign exchange spot and forward trading and the limits set. It also involves Reuters, EBS and CWS systems and the limits maintained in them.

2.4 Country Risk

Country risk pertains to the risk of business loss due to problems arising in a specific country. These problems usually stem from political and economic instability which cause countries to default on prior investment deals. Businesses often have their own definition and methodologies for calculating country risk and therefore are subject to different country risk exposures.

Country risk can often times be broken into "micro" and "macro" risks. "Micro" risks are often associated with a single firm or direct investment. For example, if a government regulation restricts a firm or direct investment making it impossible to turn a profit then there is a potential for financial loss. Country risk incorporates the probabilities of these types of losses in foreign investments. "Macro" risks refer to macro-political and macroeconomic events such as wars, revolutions, and large scale economic crises. These risks are of major concern for any global banking firm and are monitored closely. Lehman Brothers is no exception to this, and is currently looking to improve their process for generating country risk reports to ensure that they are receiving current and accurate information to help monitor their own country risk levels.

3 Methodology Part 1

This section discusses the various methods by which we completed our project. We used several techniques in helping to create a more efficient and accurate method of reporting country risk at Lehman Brothers. First, we became familiar with technical aspects related to the current processes through existing documentation. Next we met with the employees at Lehman Brothers to discuss what reporting systems are currently used, along with the systems' strengths and weaknesses. Once we established an understanding of the current process for reporting country risk we began working with the existing development team in the early stages of reforming and improving the country risk reporting system.

3.1 Learning About the Current Process at Lehman Brothers

In the initial phases of our project we read over several reports outlining the current procedure for creating the reports. To support this we conducted several interviews that helped us to form a foundation of knowledge about the proceedings at Lehman Brothers. Interviews were used both formally and informally throughout the project. Interviews were combined with observational analysis and other forms of research in order to help recognize problems in the report creation process. From this knowledge we were able to identify several different approaches to improving the formulation of country risk reports at Lehman Brothers.

3.1.1 Interviews

Through the use of unstructured interviews we collected first hand information about the steps that are taken to the create reports. Interviewing experts in the process helped us better understand the procedure and allowed us to evaluate weaknesses in the process, and provided us with ideas for improvement. We met with a variety of employees in New York. We primarily met with users of the reports, employees who work to generate the reports, and the employees that change the existing reports.

3.1.2 Process Analysis

The use of observational analysis was used in conjunction with our interviews. After discussing many areas of concern with Lehman Brothers employees we were able to identify specific issues we should focus on during our evaluation of the country risk report creation steps.

3.1.3 Evaluation Criteria for the Current Process

The next step in analyzing the development of country risk reports at Lehman Brothers was to look at each step in the process. Breaking down each step allowed us to recognize deficiencies in the system. We were then able to evaluate the relationships between the inefficiencies and steps necessary to create the current procedure. Some of the major areas of focus when analyzing the current reporting system include:

- Duplication of effort
- Unnecessary steps
- Sources of delay
- Reactive versus proactive error corrections.
- Flexibility in the system
- Data accuracy

3.2 Analyzing Product Types

A preliminary aspect of the project was to create a tool to help develop an understanding of products offered by Lehman. This was done through the creation of a data dictionary which provides descriptions of many of the fields found in Lehman databases. Another facet of the project relied on improving the accuracy of the data going into the country risk reports. Several products offered by Lehman are currently being classified incorrectly. The data analyzed during this project was done primarily for the cross currency swap products found in the HJM and SUMMIT(SUMM) source systems. This was done through the use of DBArtisan, a database access tool.

3.2.1 Creation of the Data Dictionary

The purpose of creating the data dictionary was to create a document that would help explain the data fields that go into the country risk reports. Currently there was no central document in place that describes the individual meanings of each data field. This creates a problem for many people working on the project who are unfamiliar with the data elements. The example below shows the type of information that was included in the data dictionary. The name `Loan_Amount_USD` would be a data field found in Lehman databases that needs describing.

Name	Database	Table	Sybase Definition	Default	Description
Loan_Amount_USD	Exposures	country_risk_finance_leg, country_risk_finance_deal, country_risk_adjustment, country_risk_noncorr	AMNT	NULL	It is the cumulative of all the leg Loan_Amt_USD values times the Asset_Risk_Factor value if the Currecny type is 'LL'. If the Currecny_Type is not 'LL', then the value is the cumulative of all the leg Loan_Amt_USD values.

The data dictionary shows the field name or piece of data `Loan_Amount_USD` in question. The information retrieved on each field name includes what databases and specific tables it is located it, which is then followed by its Sybase definition `AMNT`, and default value setting which in this case is `NULL`. Finally, it provides a description of what data values it can hold and a brief definition. This was done for approximately 300 data fields. Most of the information contained in the data dictionary already existed in various documents and reports. To create the dictionary we ran several queries in DBArtisan to find what data fields were found in each table and which database they are located in. The descriptions were pulled mainly from a series of reports that contained much of the information needed to define each data field. After this information was incorporated into the dictionary we met with several employees who are familiar with the data who helped fill in a number of the remaining gaps.

3.2.2 Product Type Analysis and Classifications

Once the data dictionary was established we were familiar enough with the data to begin analyzing specific product types. Research done prior to our project outlined several areas of concerns with misclassified, incorrect, or missing data. Cross currency swaps were known to have some of these issues. We used this as a starting point and took a closer look into the cross currency products to find exactly which pieces of information were being represented incorrectly. This analysis was performed by running a series of SQL queries using DBArtisan. The sequel queries allowed us to sort the data in various ways which made it easier to evaluate large amounts of information. The data was then imported into spreadsheets for further analysis. We checked for several key patterns and ways to differentiate between the different product types.

3.3 Requirements

The current process of reporting country risk at Lehman Brothers is very tedious. When approaching this project, we divided our tasks into various stages. First we had to understand the current reporting process. Reading and understanding existing documentation and doing process analysis helped us gain more knowledge about existing process and outline aspects that needed to be improved. Once we understood the current process and requirements, we met with users to come up with a general design for the prototype. The next step was to design and implement the prototype. This was done so the users could get the feel of the future system and change any requirements before the final version of the system is built. After the prototype was designed, we presented it to the users, collected the feedback, and made appropriate changes to the prototype design. Once the modified prototype was completed and approved by users, we started the implementation and testing of the system.

Defining requirements are crucial for the success of a final system. Without clearly defined requirements, design and implementation of a system would be an eternal process of requirements redefinition. To gather the requirements for our system, we looked at previous reports that were generated manually. We designed and developed a prototype to emulate the "look and feel" of the manual reports. We also spoke to current users of the report to gather functional requirements. The input from these users was also combined into the design of the prototype. In order to be able to accomplish this task, we developed the prototype system which served as the basis for the final design of the system.

3.3.1 Design and development of prototype

Once we developed a clear understanding of the current process, we designed the user interface. Designing this interface defines how data would be displayed. This was one of the most important aspects of this project. We needed to find a way to collect only the relevant information which was needed by the users and display it in the logical way. We designed the prototype based on spreadsheets that were used for current reporting. Once the design of the prototype was complete, we began implementing it. We designed the entire prototype using HTML and JavaScript. Even though it was not functional, it served as a basis to collect user requests and responses as feedback. Making a simple design gave us extra time to communicate with users and make appropriate changes to the prototype to meet the changes in needs.

3.3.2 Meeting with users and collecting feedback

After completing the prototype, we conducted several meetings with current users to collect their initial thoughts and collect feedback about the system. Through these meetings, we gained

additional knowledge about user needs and areas that needed improvement. We raised the following topics to gather as much feedback as possible:

- Current data layout
- Data completeness
- Data quality
- Ease of use
- Areas of improvement

3.3.3 Re-design prototype

Through the preceding interviews, we collected valuable user feedback and documented it. Additionally, we outlined the sequence of steps required to be able to improve the prototype. The next step was to make the changes to the prototype. The design of the prototype was modified to reflect the changes from the user feedback.

3.4 Architecture and design

After completing the prototype and collecting all requirements for the project, we started on the architectural design, which included the software design and data modeling. The system was designed using the Java 2 Enterprise Edition (J2EE) component based approach. In a J2EE application, views are Java Server Pages (JSP) files which generate HTML pages. The "view" is the user interface or the screens that the application user actually sees and interacts with.

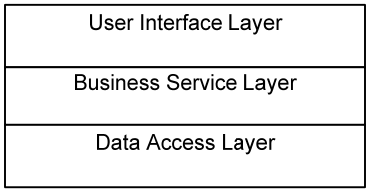


Figure 3-A - System architecture layers

The system architecture consisted of three major layers. They are user interface layer, business service layer, and data access layer. User interface was developed using Spring model-view-controller (MVC) framework. MVC is the pattern that helps to separate presentation from business logic. It is responsible for interpreting the user's request and interacting with the application's business objects in order to perform the request. We also utilized Spring Singleton framework for business service implementation and validation. For Data access layer we utilized Hibernate for Object/Relational Database mapping. This has greatly reduced the amount of code that is required for database operations. We also kept the ability to use Java Database Connectivity (JDBC)/Stored Procedures in the data access layer for any database operations that are too complicated to map via Hibernate framework. After designing the architecture we made a few minor changes to the database

structure. Due to ongoing business requirements analysis, we could not complete the final design of the database. We did, however, make several improvements or changes like cleaning up the database field names, adding time-specific information such as date/timestamps and user created or modified by, and incorporating improved user security characteristics.

3.5 Development

Once the requirements were completed for each separate part of the system were finished, we began the implementation of the application layer. Some of the parts were in the process of being approved by the end users and upper management. We started the implementation of the Administration section since most of the requirements were complete, which served as the basis for all other parts of the system including data and user management. The front-end interfaces were implemented in JSP.

3.6 Development testing

We tried to follow an iterative process throughout the development cycle by testing while implementing. After developing the various parts of the application, we went through several cycles of testing. Next we addressed the problems that arose and made necessary revisions. Once these changes were completed, the system was error-free.

4 Results Part 1

This section provides the results of the data analysis and the development of the country risk reporting system. Using the methodology described in the previous chapter we were able to develop a functional prototype to model the future country risk reporting system. Furthermore, we were able to assist in some of the data quality issues facing the country risk reports.

4.1 The current process

The current system for reporting country risk at Lehman Brothers involves many manual processes. The data used to create the reports comes from a variety of source systems which is aggregated into a series of databases. Next, a group of analysts go through a progression of steps to create the country risk report. The country risk reporting process is comprised primarily with manual processes which have proven to be labor intensive and time consuming. It takes several employees two to three days to complete the final report, and thusly the reports can only be produced bi-weekly. This creates a problem since many of the financial statistics in the country risk report are changing daily. The following flow chart shows the current steps taken to produce the country risk reports.

Error! Objects cannot be created from editing field codes.

Figure 4-A – Flow chart diagram of current process

4.2 Data results

A major step in improving the country risk reporting process relies on validating the accuracy of the data going into the reports. Many of the items found in HJM and SUMMIT source systems are classified as swap deals, when there are actually different types of products. The SUMMIT and HJM source systems provide swap related financial data in the cross currency swap tables. These were the only source systems that were considered when analyzing the swap, swaption, exotic and null product types since they include emerging market country data, which is the primary data source for the current country risk reports.

4.2.1 Misclassified Product Types and Incorrect Valuations

The current problem with the swap data is that the same methodology is being applied to different products. The cross currency swap calculations are being used for each of the following product types: swaptions, exotics and IR swaps. In this analysis we tested SUMMIT and HJM to see if all the swap products were cross currency swaps. As predicted we found that most of the products are in fact not cross currency swaps.

In the evaluation we broke down each data field into several components. We first looked to see what values produced could be used for segregation, meaning could the data values found in a particular field be used to sort the information. Next, we looked at the values in each field and discovered whether or not they could be used for valuation. In other words could the value in each field be used in mathematical calculations for computing risk for each particular product type. For example, the `curr` data field stands for currency type. This information is used in calculations based on the product in question. Data fields such as `FocusId`, which is a sequence of letters and numbers that uniquely identifies different securities, are not used for valuation purposes.

We also looked at several patterns in the data to try to discover different ways to group the information. From this we were also able to recognize any anomalies in the data. In addition, we found that there were a number of fields with missing data. This information was recorded in a separate spreadsheet for further analysis. Finally, we ran separate SQL queries for each data field to take a closer look. This was done to count and sort the data. We recorded what types of values each field contained and a count of each value that appeared. This analysis was carried out for each of the 70 data fields found in the `dmsExtract` data table for both the SUMMIT and HJM source systems. The following shows an example of the analysis ran for several data fields. Refer to Appendix 1-B: Data Analysis Results for complete data analysis results.

Field Name	use for segregation	Use for valuation	comment	example 1 (IR Leg 1)	Example 1 (IR Leg 2)
size	Maybe	yes	vary from 9000 to 10,000,000,000	25000000	25000000
initNotl	Maybe	yes	0.0(242)	25000000	25000000
bookid	Yes	no	majority 60360(1879)	71547	71547
cpty	Maybe	maybe	cpty = cpty_name	052897HYPO	052897HYPO
cpty_name	Maybe	maybe	see cpty	052897HYPO	052897HYPO
eff	No	maybe	majority 20051123(248)	20050223	20050223

Table 4-A - Sample data analysis

The results were then used to identify specific problems with the cross currency swap data for later correction. Running similar examinations of all data types in question will help to ensure the information going into the reports is accurate.

The country risk report includes only countries that are considered to be emerging markets. Once the analysis was complete we found that the HJM source system contained a limited amount of data in question. The only swap type product found in HJM for emerging market countries were (NULL) products. The SUMMIT source system was found to have several different products that are currently misclassified. In the SUMMIT system we found swaptions, swaps, exotics, CAPTR, and FRA being classified as cross currency swaps when several of them are in fact not.

4.3 Design database structure to meet the needs

The database was redesigned to reflect the changes made by business analysts based on the prototype and other requirements previously described. The following tables were created to store relevant information.

country_risk_bond_type	
This table stores information about different products.	
ID	Unique identifier for bond entity; generated by database server.
bond_type_name	The name of the bond entity.
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_countries	
This table stores information about all countries in the system.	
ID	Unique identifier for country entity; generated by database server.
Country_name	The name of the country entity.
ISO_2	The International Standard of the entity
ISO-3	The International Standard of the entity
ISO_number	The International Standard number of the entity
watchlist	The watch list flag
pegged_currency	The pegged currency flag
elp_limit	The estimated loss potential limit of the entity
coverage_id	Foreign key referencing coverage ID
region_id	Foreign key referencing region ID
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_country_tier	
This table stores correlations between markets, ratings, and liquidity.	
ID	Unique identifier for country tier entity; generated by database server.
rating_equilavent_id	Foreign key referencing coverage ID
liquidity_id	Foreign key referencing liquidity ID
Market_id	Foreign key referencing market ID
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_coverage	
This table stores information about different risk coverage types.	
ID	Unique identifier for risk coverage entity; generated by database server.
coverage_name	The name of the risk coverage entity.
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_liquidity	
This table stores liquidity information in the system.	
ID	Unique identifier for liquidity entity; generated by database server.
liquidity_name	The name of the liquidity entity
description	The description of the liquidity entity
Code	The code of the liquidity entity.
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_market_duration	
This table stores information about market durations in the system.	
ID	Unique identifier for market duration entity; generated by database server.
maturity_code	The maturity code for duration entity
description	The description of the liquidity entity
maturity_bucket	The maturity bucket of the liquidity entity
start_day	The start day of liquidity entity
end_day	The end day of liquidity entity
Market_id	Foreign key referencing market ID
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_markets	
This table stores information about different markets in the system.	
ID	Unique identifier for market entity; generated by database server.
Market_name	The name of the market entity
Code	The market code
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_rating_equivalent	
This table stores information about different rating types in the system.	
ID	Unique identifier for rating entity; generated by database server.
tier_name	The name of the tier entity
ICR	
Moody	Index
SandP	Index
tier_code	The code of the tier
last_modified_date	Date of last change
last_modified_user	Last modified user

country_risk_region	
This table stores information about regions in the system.	
ID	Unique identifier for region entity; generated by database server.
Region_name	The name of the region entity
correlation_factor	The correlation factor of each region
last_modified_date	Date of last change
last_modified_user	Last modified user

4.4 Create Interface Mark-up and Get Feedback

An interface prototype was initially done in basic HTML and JavaScript. A screen shot of this can be seen below in Figure 4-B.

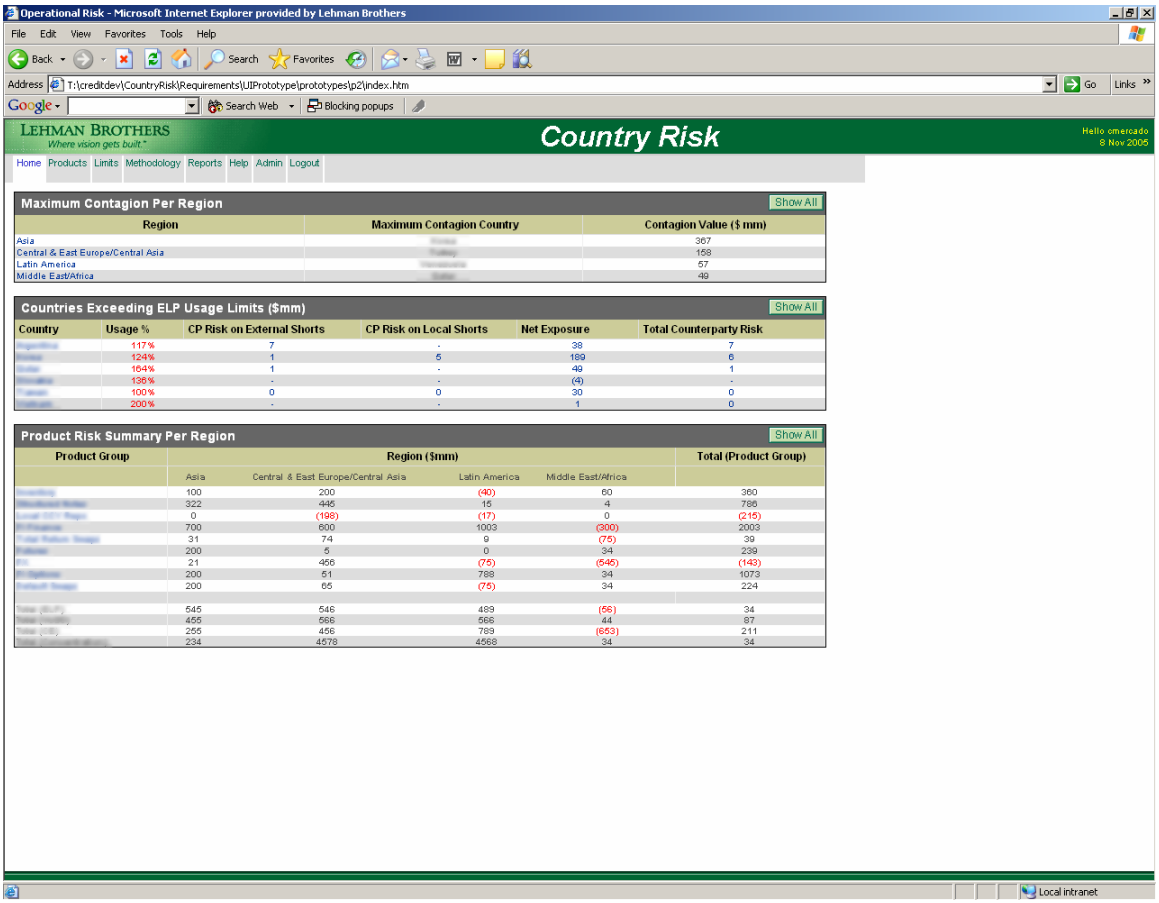


Figure 4-B – Home screen of the Country Risk Reporting interface

A user receives the most important information on the home page, which includes countries with the maximum contagion for every region, a list of countries exceeding ELP Usage, and product risk summaries per each region. In addition to this information, the user has an ability to view information for all countries and a drill-down to each country or product group.

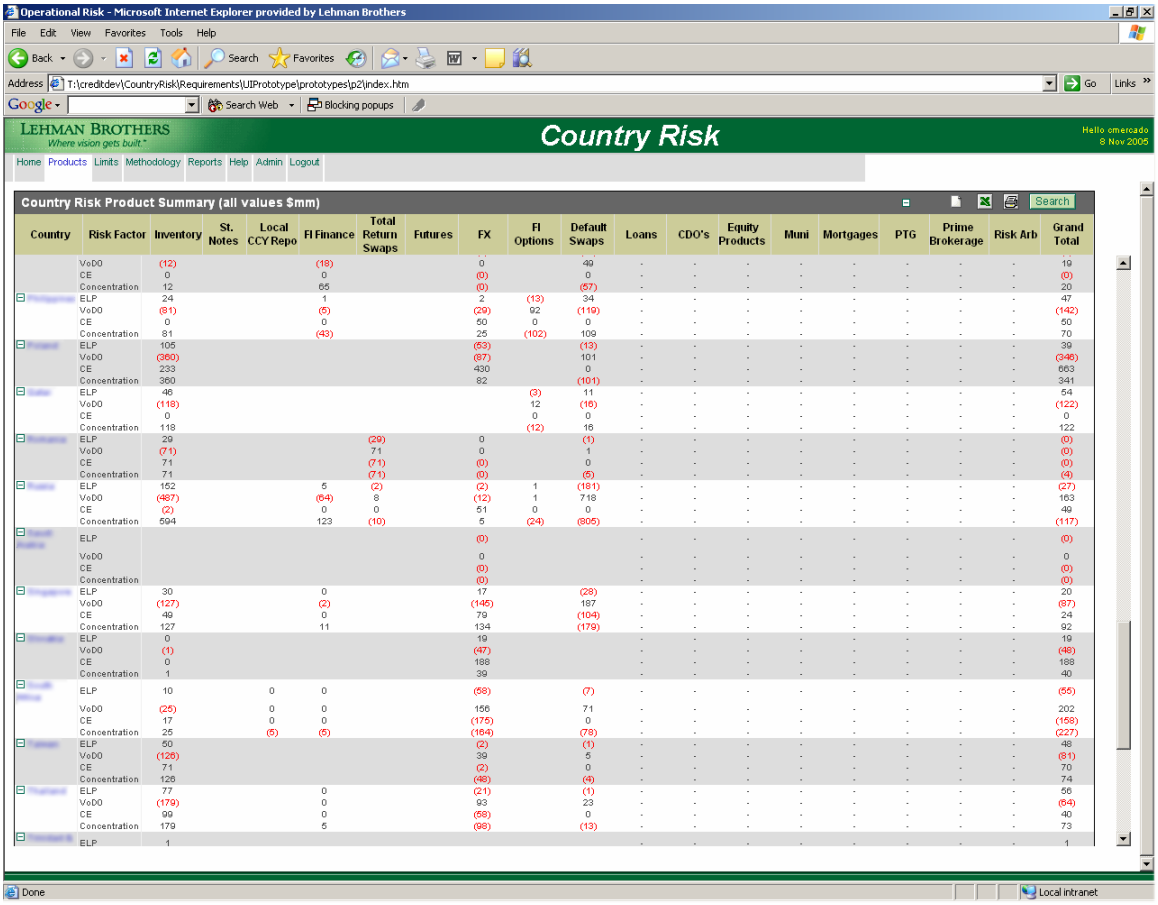


Figure 4-C – Products screen of the Country Risk Reporting interface

From the products page, the user has an ability to look up product summaries for each country.

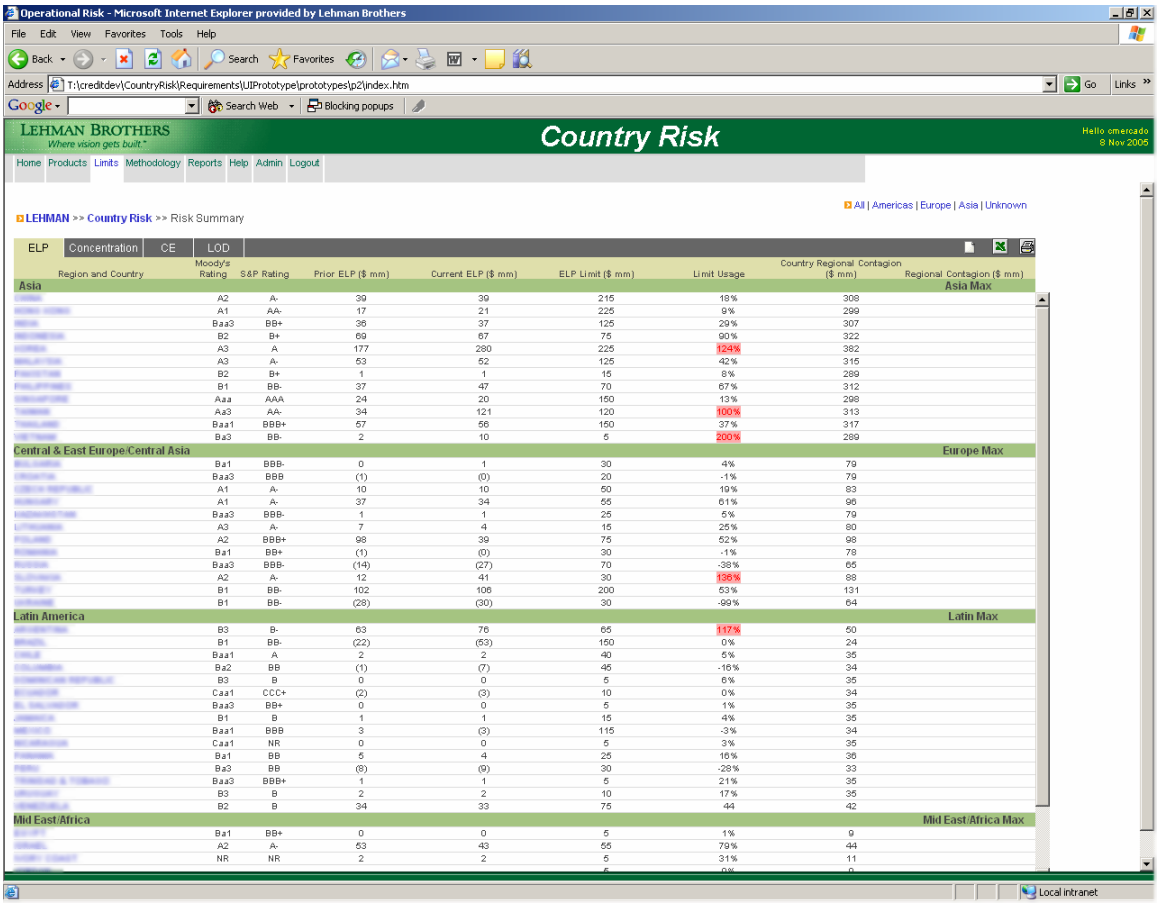


Figure 4-D – Limits screen of the Country Risk Reporting interface

The user can view limits information for each country on this screen including ELP, Concentration, CE, and LOD. Also, user has an ability to click on each country and look up the products information for each country.

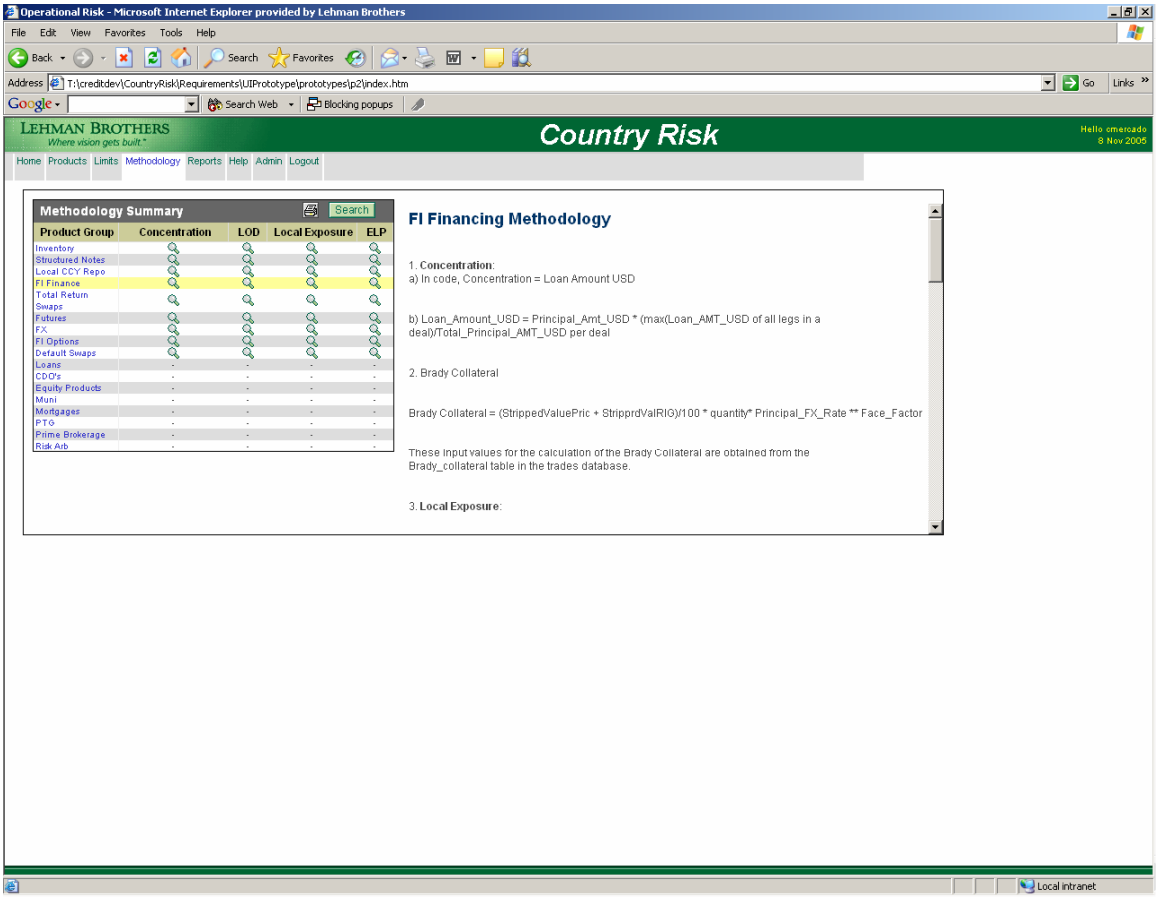


Figure 4-E – Methodology screen of the Country Risk Reporting interface

The user has the ability to look up the methodology for each product group.

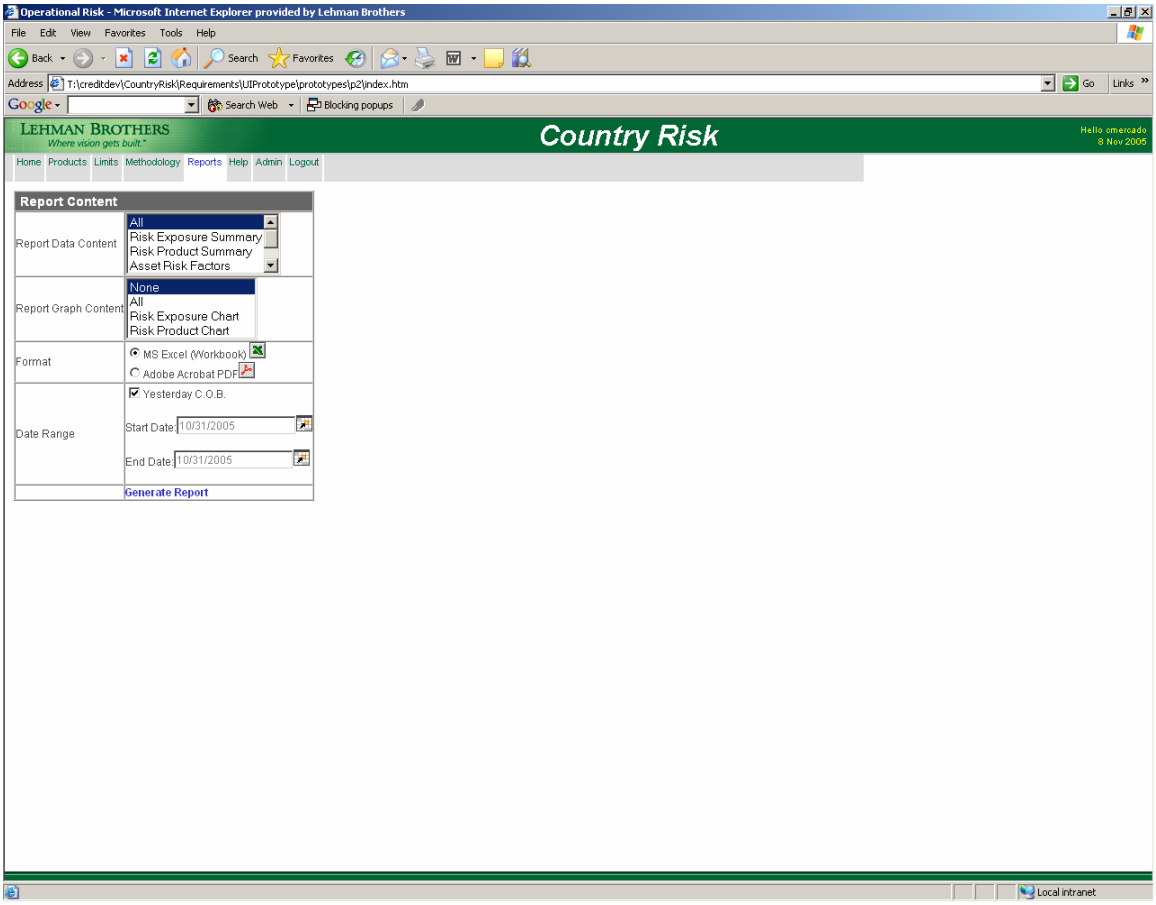


Figure 4-F – Reports screen of the Country Risk Reporting interface

From this screen, the user has an ability to configure and create MS Excel or PDF reports. The user can select the data range for the reports.

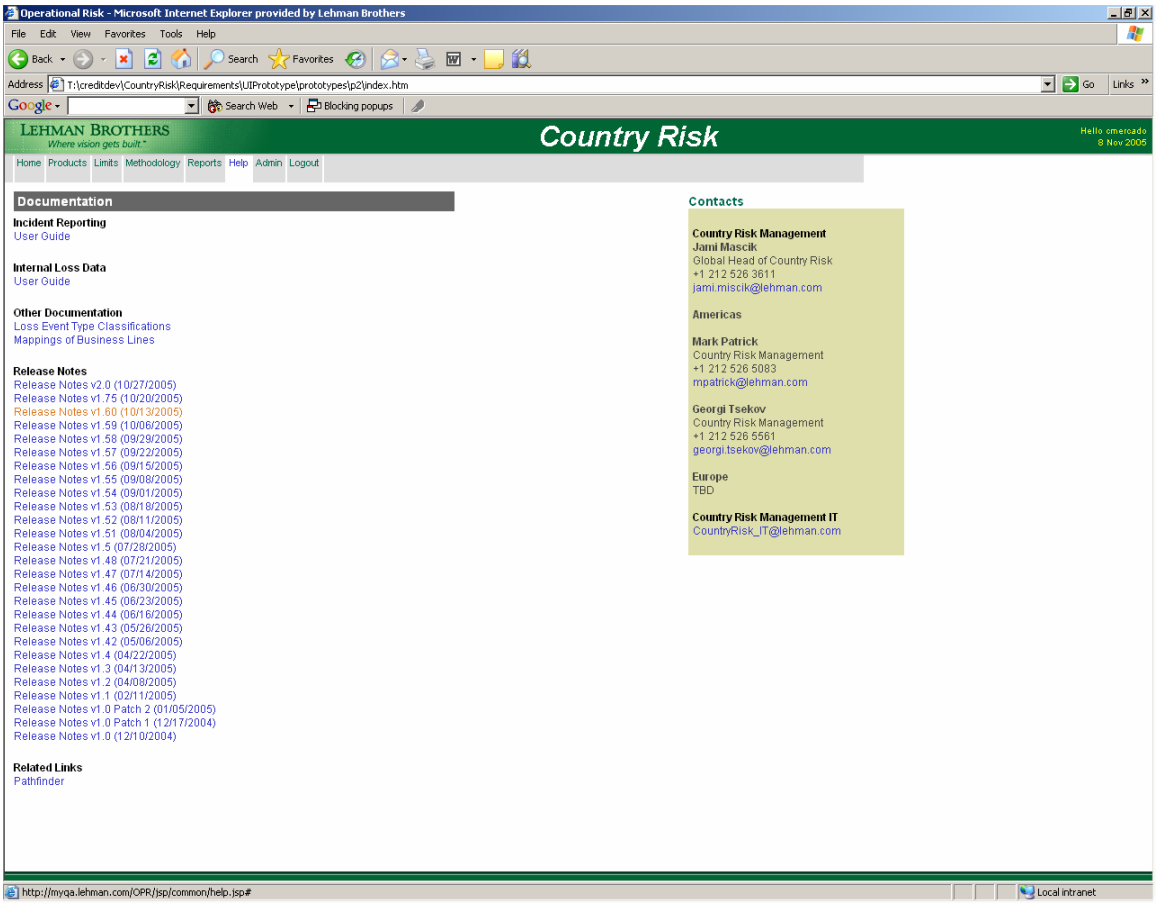


Figure 4-G – Help screen of the Country Risk Reporting interface

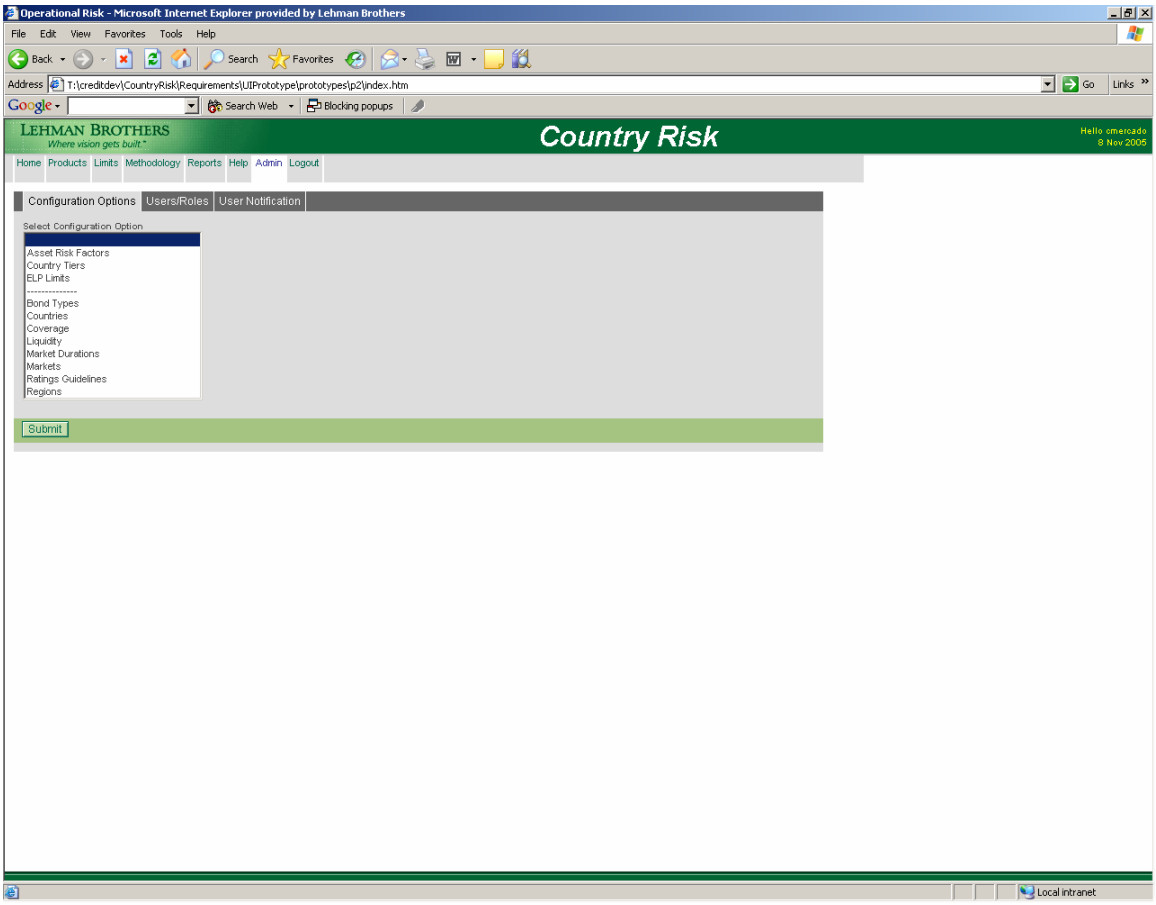


Figure 4-H – Admin screen of the Country Risk Reporting interface

The above screen allows users to configure data for the report throughout the user interface. Only administrators have access to the screen. They have an ability to modify country, bond type, coverage, liquidity, market duration, markets, regions, and rating guidelines information. After changing the information, administrators can generate a read-only report which will incorporate the changes made. In addition, they have an ability to modify user account information and user notification information from the Administration screens too.

4.5 Implement the System

We divided the implementation of the system into three sections: setting up architecture, implementing the database structure, and implementing the application layer(s). The implementations of these sections were done with Lehman Brothers database and application infrastructures in mind.

4.5.1 Setting up Architecture

The system architecture consisted of three major layers. They are user interface layer, business service layer, and data access layer. We incorporated Spring model-view- controller (MVC) framework into user interface layer and also utilized Spring Singleton framework for business service

implementation and validation. For Data access layer we utilized Hibernate for Object/Relational Database mapping. We also kept the ability to use Java Database Connectivity (JDBC)/Stored Procedures in the data access layer. Each of them has advantages and disadvantages, that is why we decided to keep both of them to make the application more robust.

4.5.2 Implement the Database Structure

Lehman Brothers technology departments generally use Sybase SQL servers. These servers use a variation of Transact-SQL as their language for describing the tables in the relational database. Due to ongoing business analysis, we could not complete the final design of the database, but we implemented some minor changes including cleaning up the database field names, adding time-specific information such as date/timestamps and user created or modified by, and incorporating user security and suggested a new scheme for future development.

4.5.3 Implement the Application Layer

The implementation of the Application Layer was based on the prototype designed and was implemented in Java Server Pages. The screen shots of the final system can be seen in Figure 4-I.

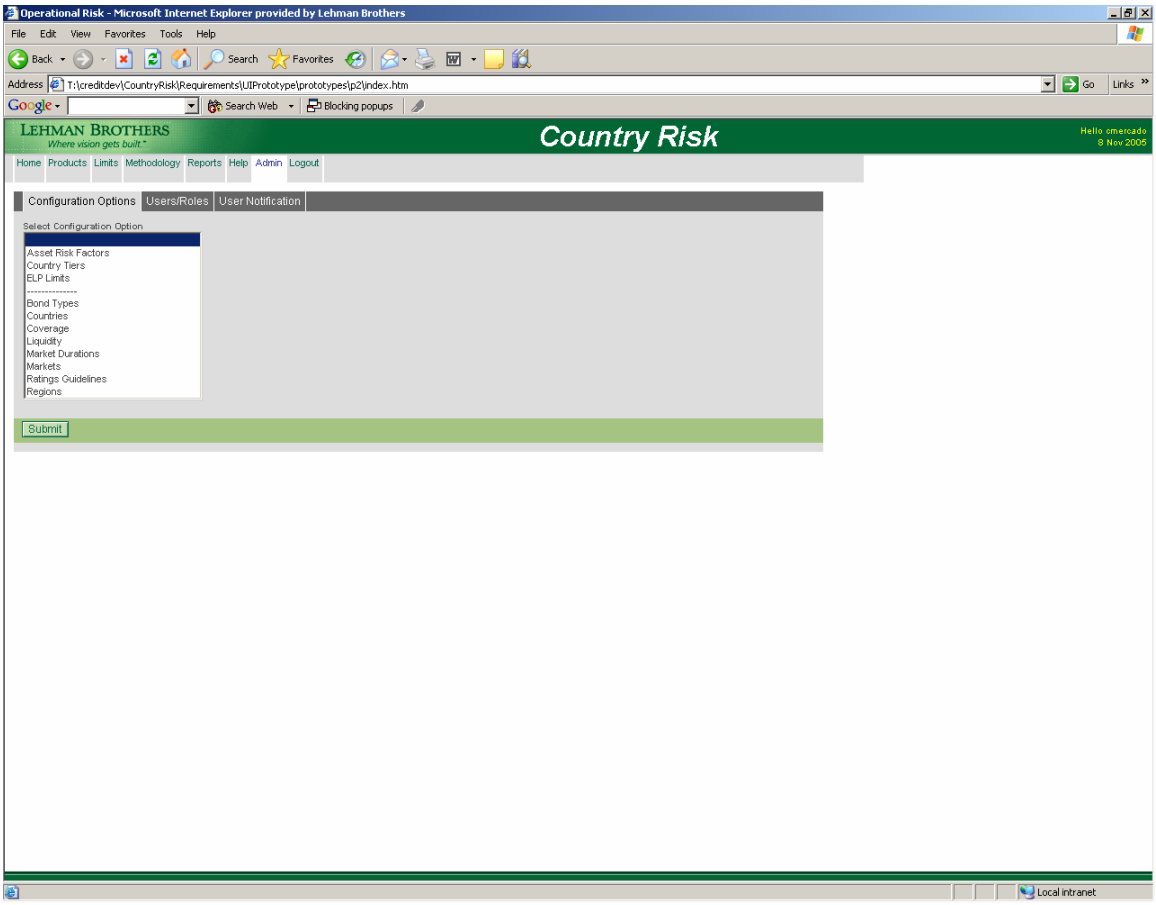


Figure 4-I – Interface screenshot: Admin

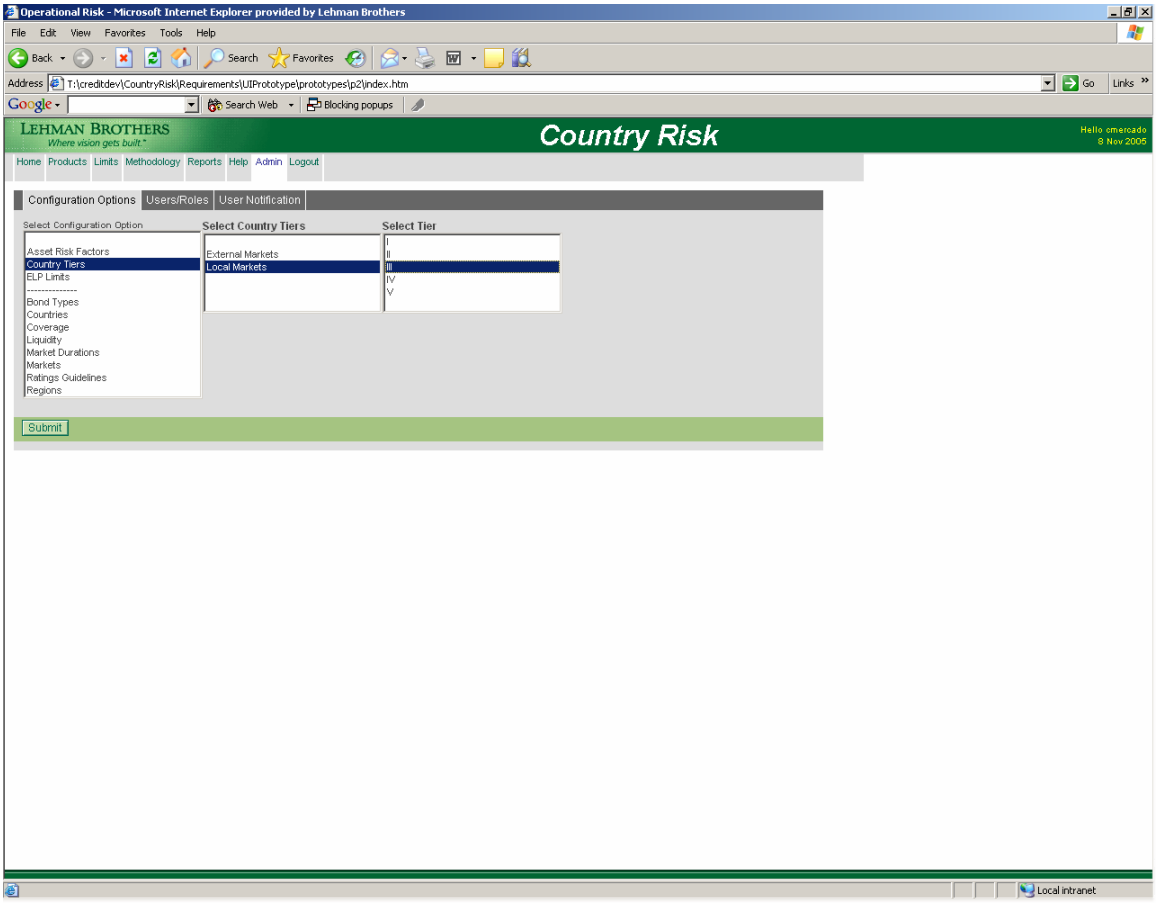


Figure 4-J – Interface screenshot: Admin

Selecting different configuration options allows Administrators to modify different aspects of the report.

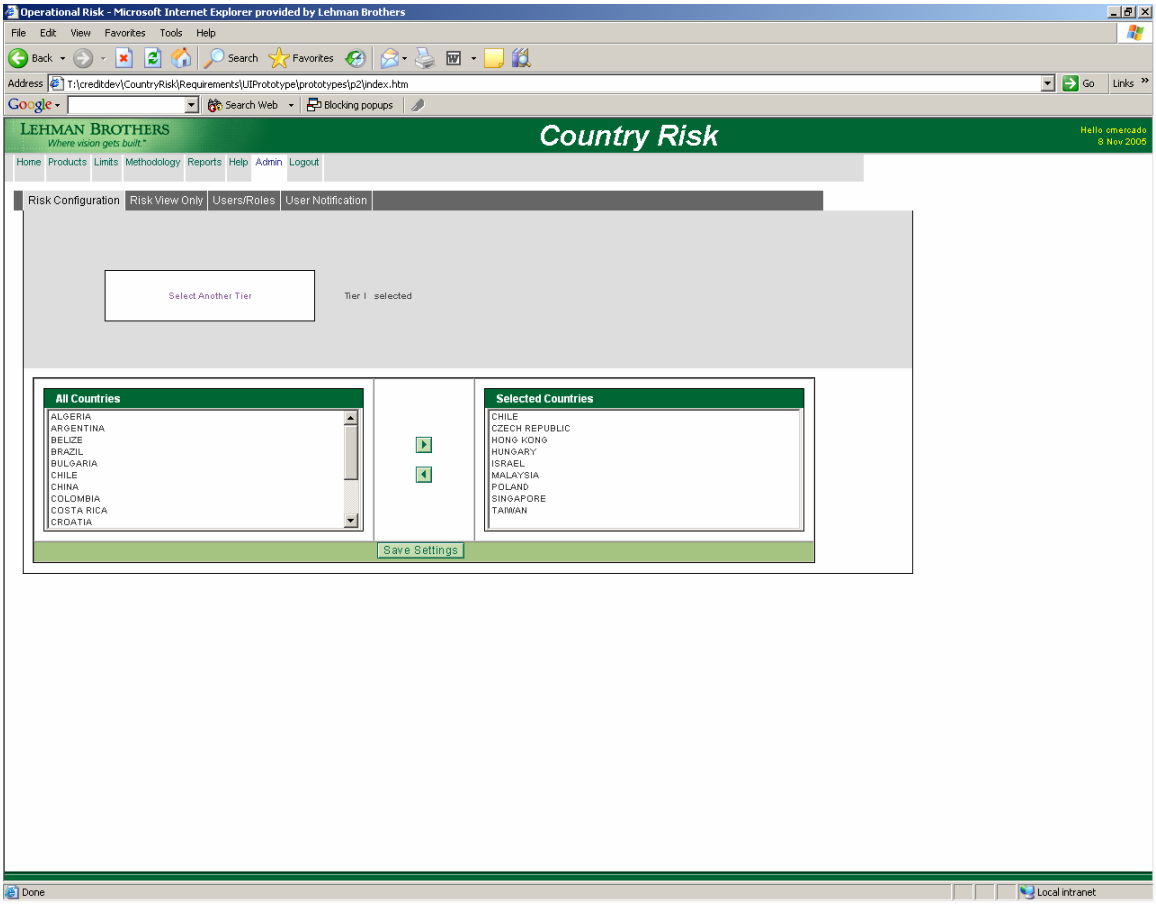


Figure 4-K – Interface screenshot: Country Tiers

Selecting different tiers allows users to modify country tiers.

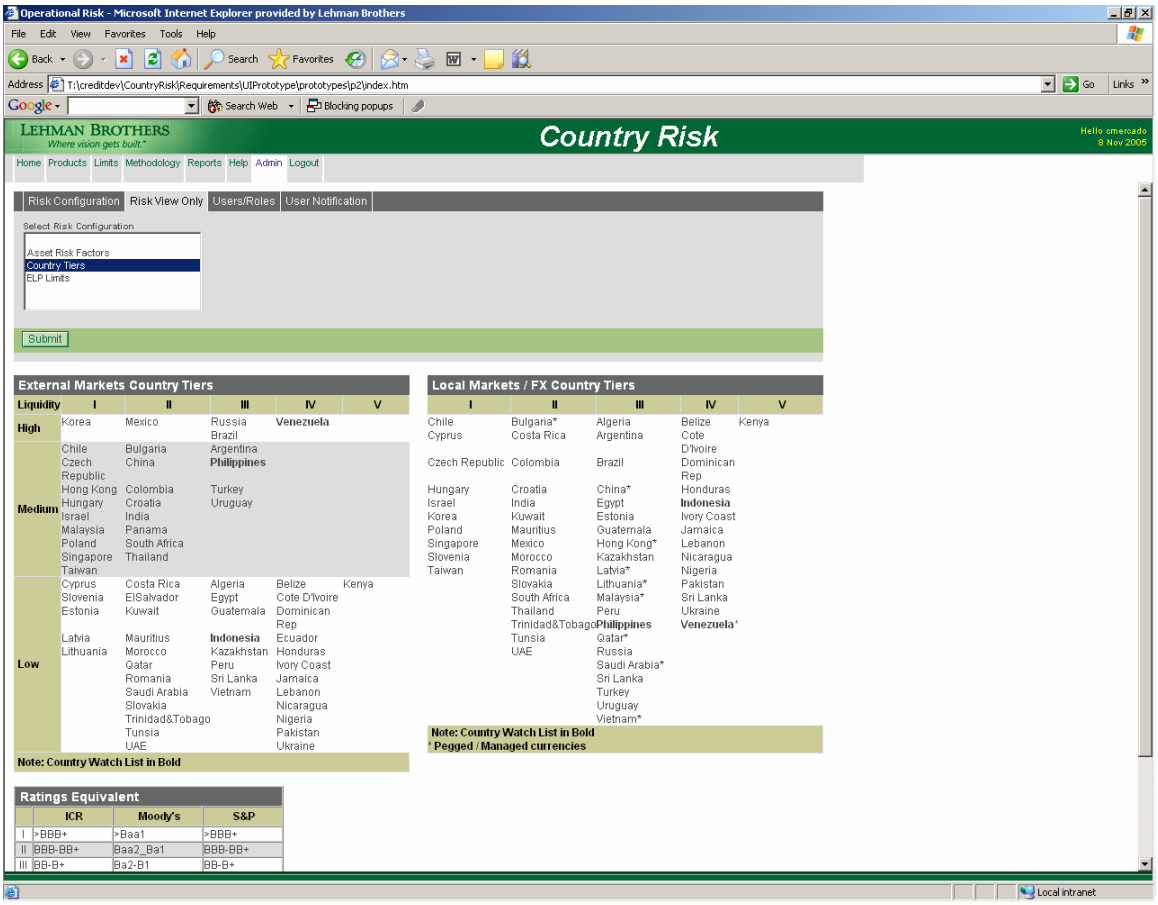


Figure 4-L – Interface screenshot: Country Tiers (Read Only)

After modifying country tiers, user has an ability to view the Country Tiers information in an easy-to-read report.

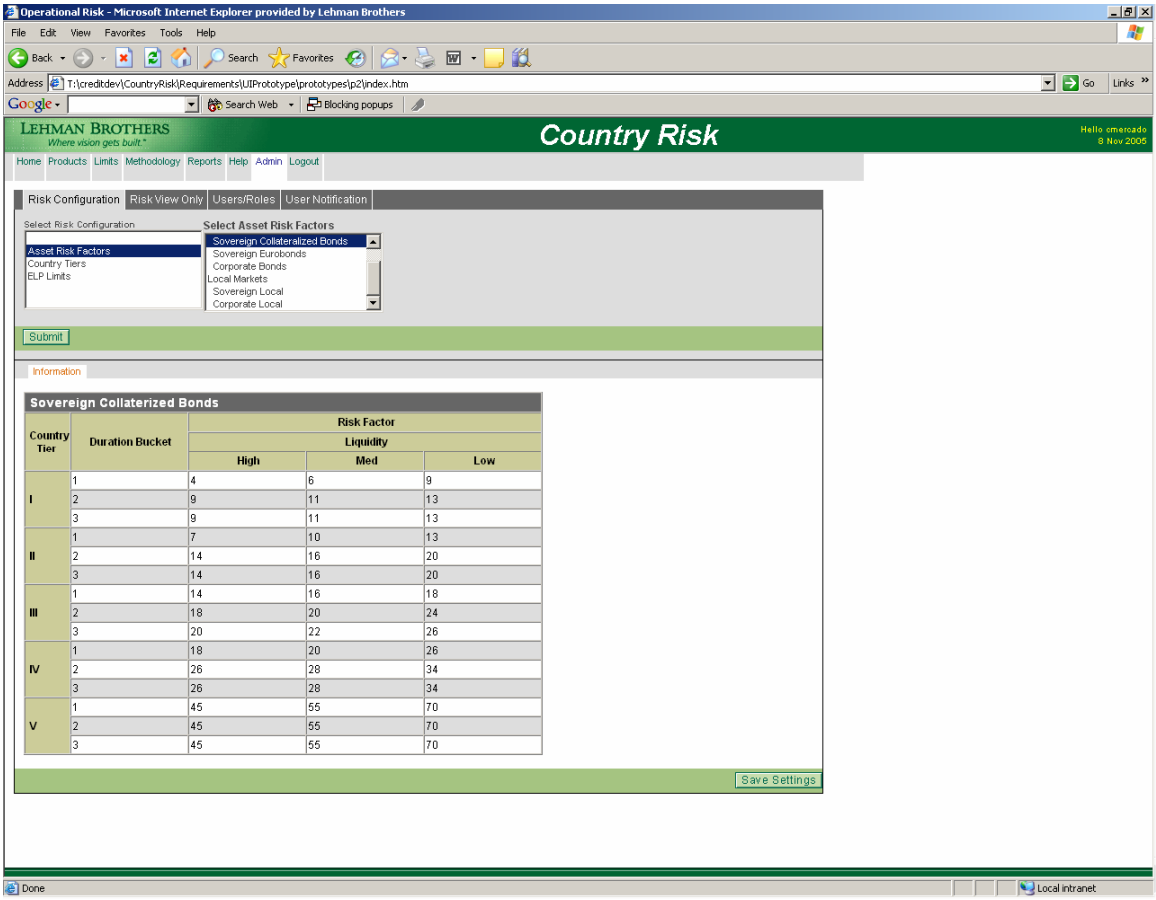


Figure 4-M – Interface screenshot: Asset Risk Factors

This screen helps administrators to modify risk factors for both local and external markets.

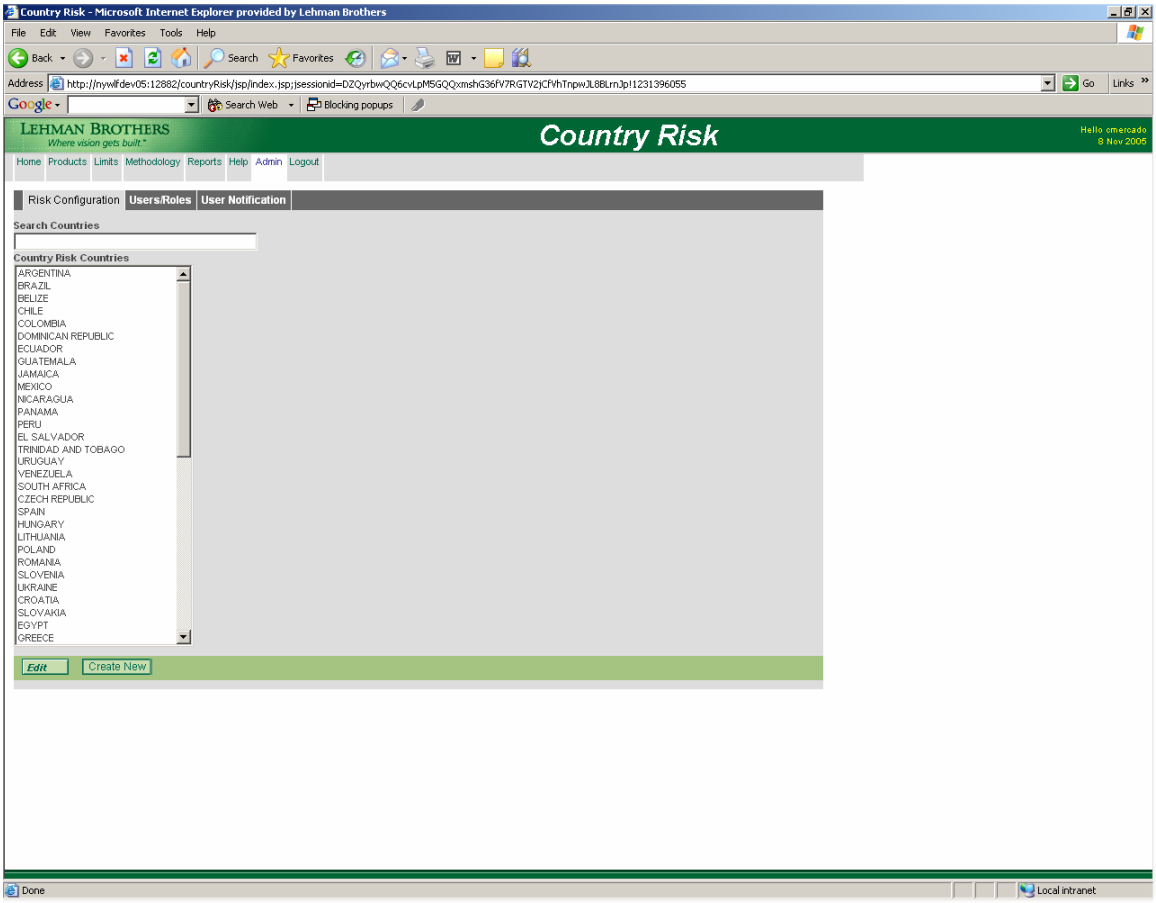


Figure 4-N – Interface screenshot: Select Country screen

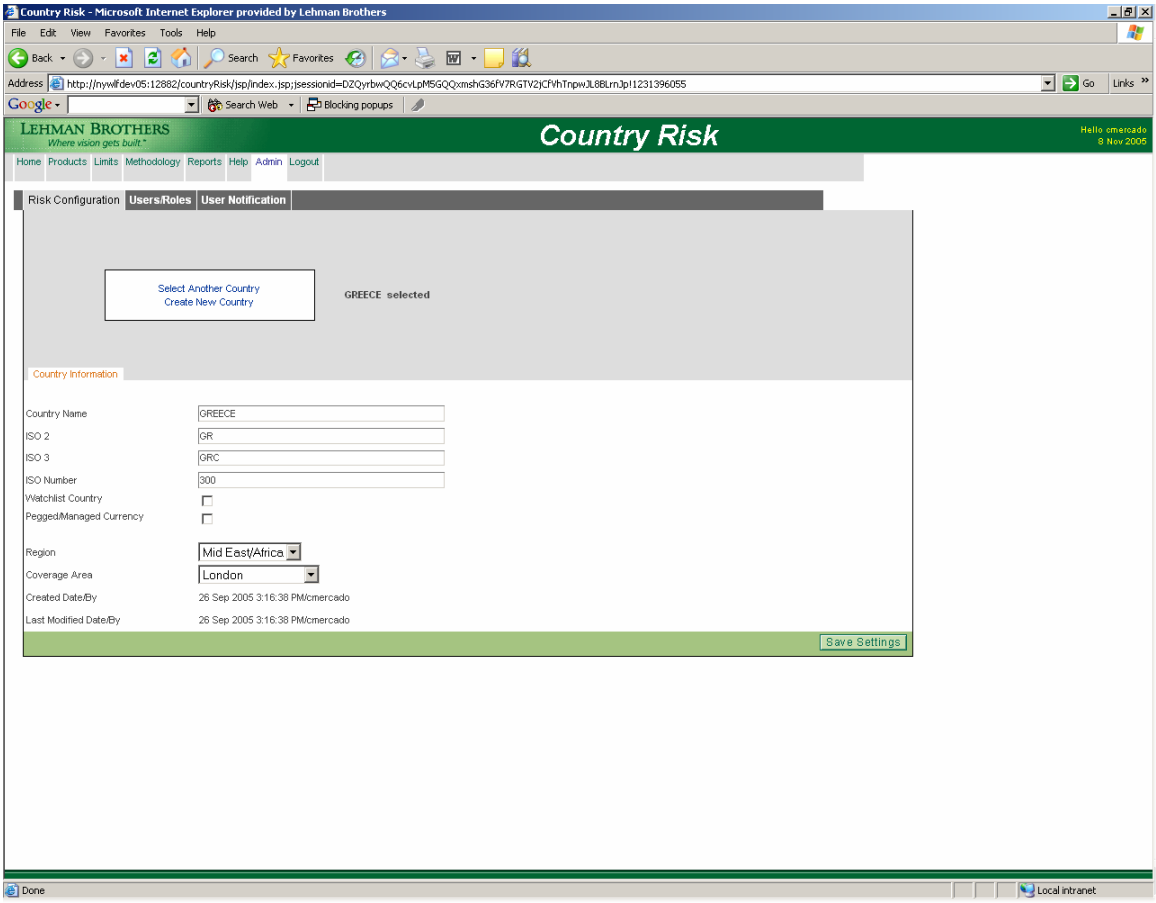


Figure 4-O – Interface screenshot: Country Configuration

This screen helps an administrator to modify and save any information about countries available in the system.

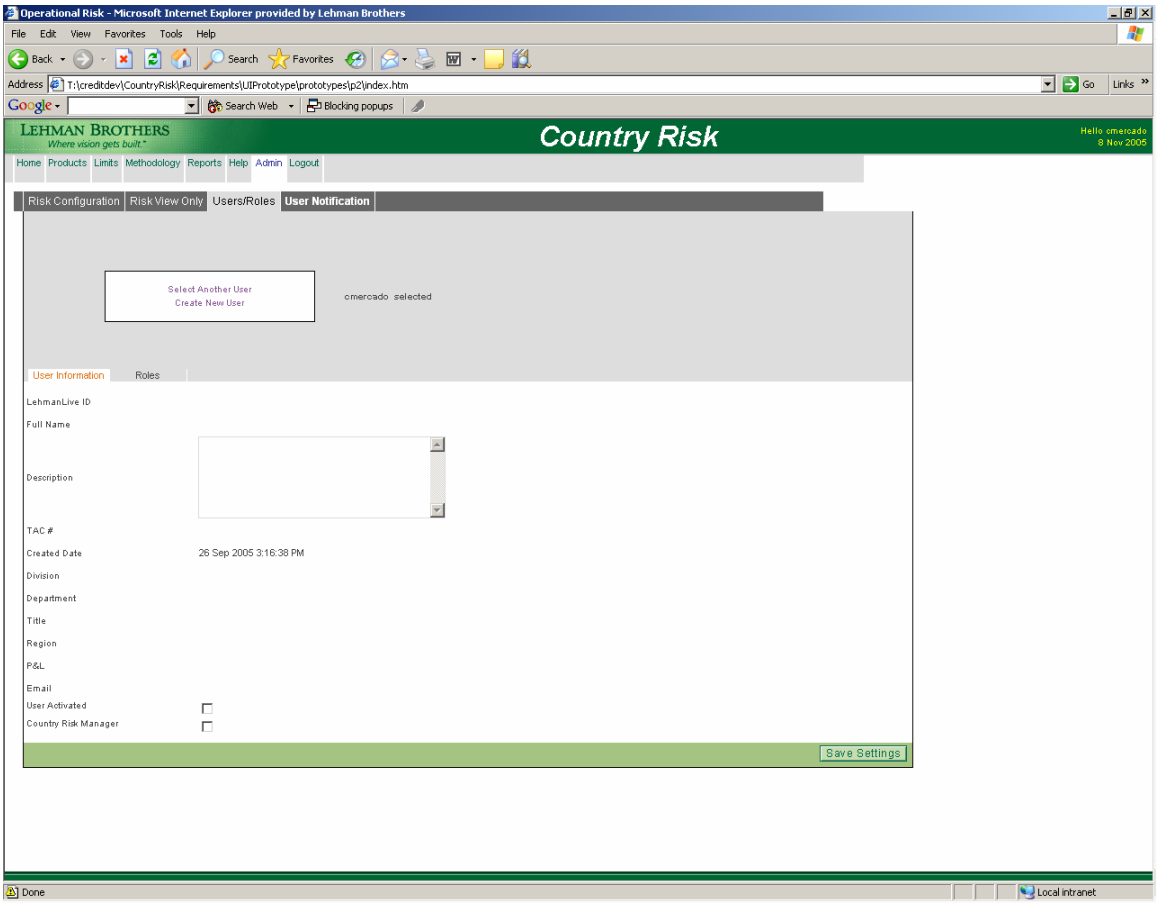


Figure 4-P – Interface screenshot: User Roles Configuration

This screen helps administrators to modify and save any information about users in the system.

4.6 Testing the system

During the system implementation we went through multiple iterations of testing. Testing of the system was done based on the requirements and the prototype that were developed. During the testing stage we went through multiple iterations of user acceptance testing too. This testing is completed by the users of the system.

5 Conclusions Part 1

The same methodologies that are still being applied to cross currency swaps are also being applied to each of the product types outlined in the previous chapter. The chart below shows a breakdown of the percentages of each product type found in the SUMMIT source system.

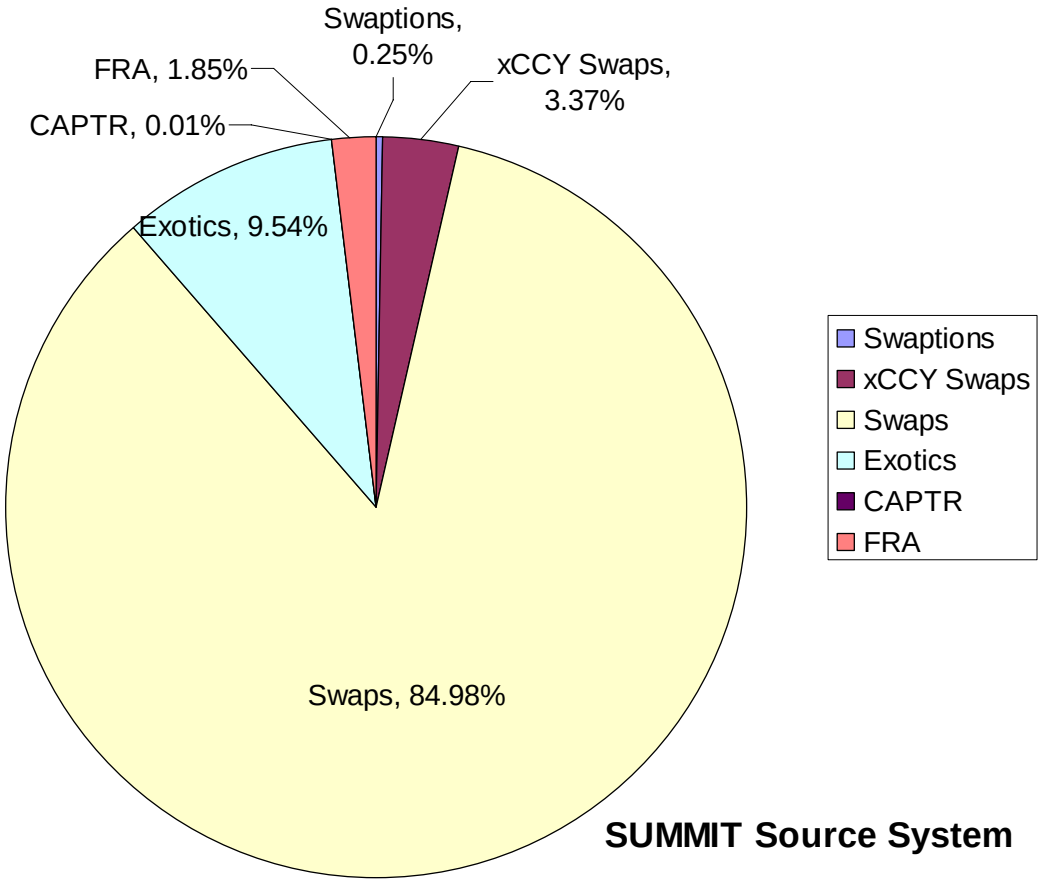


Figure 5-A – Summit source system breakdown

Using the cross currency swap methodology to calculate each of these different product types limits the accuracy of the risk calculations for each product. Since the mythologies currently do not represent the differences in these products ELP limits, and other country risk statistics are misrepresented in the reports.

In the HJM system we found only with (`NULL`) swap related product types for emerging market countries. However, the methodology being applied to this still may not be representative of the product type.

Continuing with the data analysis we then looked at the `country_risk_fx` table or commonly known as the cross currency swap table. This table is the primary country risk table which is comprised of all the financial information required for the country risk reports pertaining to cross

currency swaps. The swap data from SUMMIT and HJM is part of the `country_risk_fx` table. However, when the data from SUMMIT and HJM are imported into the cross currency swap table some of the data describing each of the product does not carry over. The most significant piece of missing information is the data field with describes the product type. As mentioned earlier, swaps, exotics, swaptions, CAPTR, FRA, and (`NULL`) were labeled as such under the data field "prodType" or product type in the SUMMIT and HJM systems. In `country_risk_fx` the "prodType" data field is not carried over. This creates another problem because there is no efficient way to differentiate these swap related product types once it is imported to the `country_risk_fx` table.

The last part of the data analysis was to so compare the different product types with each other. Since the product names are not carried over into the cross currency table it was important to see if there other ways to distinguish between them after they are imported. However, we found that it many cases you can not easily spot one product type from another. The following is an example of the analysis comparing each product type using only the data available in the `country_risk_fx` table. Please refer to Appendix 1-C for the complete analysis.

	IR Swap		Swaption		xCCY swap
Cntry_Rsk_ID	14752	14753	31421	31422	16572
Product_Classification	fxs	fxs	fxs	fxs	Fxs
Source_Deal_Ref	100404L	100404L	1034301L	1034301L	180184E
Source_Leg_Ref	100404L.63780	100404L.63781	1034301L.763N6T	1034301L.763N6V	180184E.186444
Portfolio	ZAR500	ZAR500	TWDFXIRSWP	TWDFXIRSWP	PLN592
Trade_Prod_ID	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
Prim_Prod_ID	1705430	1705430	1705430	1705430	1705430
Cpty_Prod_ID	1705439	1705439	1705354	1705354	1705467
Source_Prim_Prod_ID	USD	USD	USD	USD	USD
Source_Cpty_Prod_ID	ZAR	ZAR	TWD	TWD	PLN

Table 5-A - Sample country_risk_fx product comparison table

Using the full version of this spreadsheet we were able to identify several defining characteristics for some of the product types. In the `country_risk_fx` both IR Swaps and Swaptions contain two legs or transactions. The xCCY swaps or cross currency swaps were more easily recognized since they have legs with two different currencies. However, IR swaps and Swaptions appear to be the same as IR swaps in the `country_risk_fx` table. This lack of distinction significantly contributes to the misclassification of cross currency swap data.

6 Recommendations Part 1

The development of an improved country risk reporting system at Lehman Brothers is a large scale project. Having accurate, first hand financial risk information on a daily basis will undoubtedly have major benefits for the Lehman Brothers organization. Even though the project is still in the early phases a significant amount of progress has been made. In order to ensure this progress through the development of the new system we have outlined several recommendations to consider along the way.

6.1 Data Analysis Recommendations

The results of the data analysis showed that several issues exist which limit the quality of data in the country risk reports. More specifically, the products in the cross currency swap table are misclassified. This means that the same methodologies are being applied to each of the different products in this data table. The first recommendation would be to decide what methodologies need to be applied to each different product. If the products are not related than the methodologies should reflect that by being adjusted. This will ensure that each products risk calculations are accurate.

The next recommendation we have is to develop a way to incorporate the product type information into the `country_risk_fx` table. This will help reduce the amount of misclassified items and provide an easier way to identify each product found in the `country_risk_fx` table. Including the product types in the country risk table will also make it easier to locate individual deals separate the data in the future.

Since the current country risk report is only comprised of risk information from emerging market countries we recommend that in the future it incorporate methodologies for all countries. Even though the risk associated with emerging countries is higher, investing in all countries carries financial risk. To ensure that the country risk reports are complete the report should represent each of the countries in which Lehman Brothers invests.

Furthermore, the data dictionary that provides glossary type information on many of the data fields found in the country risk report should be incorporated into the new reporting system. This will allow end users to search definitions and locations to each data field found in the final reports. It will also be a useful future tool for the development team to familiarize themselves with the data going into the country risk reports.

Next, we recommend that the data analysis carried out be duplicated for each of the products or deal types in the country risk reports. This will ensure that all the data is classified

correctly in the future and that methodologies can be create for un-related product types, thereby increasing the overall quality of data in the country risk reports.

6.2 Technical Recommendations

Finally, there are a number of recommendations we would like to offer on the technical side of the project too. First, we recommend that the development process needs to be defined and documented in the early stages of the process. In addition to that, the development should continue in this prescribed iterative manner. This will save a lot of time and labor when each part is implemented into the system. This will allow others to continue working on the system even though the final user requirements may not be established. Lastly, we recommend opening communication lines with the users upon completion of each step of the development process to confirm progress and gain feedback for improvements.

7 Methodology Part 2

The process of checking foreign-exchange (FX) spot credit limits across multiple trading systems, both internal and external, at Lehman Brothers is a tedious task. When approaching this project, we divided our tasks into various stages. We first had to understand the current process. Once we understood what was involved with the current process, we then brainstormed and designed the new process and how it could best be executed. Once the new process was designed, this system needed to be implemented and integrated in with the existing infrastructure. Throughout the preceding steps, we kept detailed notes. To consolidate and formalize these notes, we needed to compile a single, coalescent document.

7.1 Understanding the Current Process and Problem

The initial phase of our project was to develop an understanding of the current process for reporting and comparing various FX credit limits performed at Lehman Brothers. Understanding the current process allowed us to define a clear purpose of the project and its importance. To gain knowledge about the existing process, we conducted several semi-standardized interviews with the current users. The interviewed included users from the following departments:

- Credit Risk Management
- Quantitative Risk Management
- Risk Technology

The interviews provided us with an overview of the process flow from different aspects. From these different aspects, the different systems involved were exposed. From these interviews, we also defined a clear purpose of the project and its importance. We then highlighted the inefficiencies and identified several areas for improvement in reporting FX credit limits at Lehman Brothers.

7.1.1 Meetings with Users

Through meetings with employees involved in the current process, we learned the details of the process. The credit and quantitative risk management departments were primarily the administrators and executors of the current process. The employees in this department were asked to similar questions to explain the following:

- Explain the current process (from their perspective)
- Their role in the current process
- What aspects they would like to see improved

From the preceding interviews, we learned of the process flow and of the various systems that are used for data sources such as Credit Work Station (CWS), Reuters and EBS.

7.1.2 Understanding Systems/Sources Involved

We learned of the various systems used for sources and reference through meetings with the current users. Of these sources, we identified CWS, Reuters, and EBS as systems that we needed to investigate further. All of the systems managed FX credit limits. CWS managed the total Lehman Brothers credit appetite while Reuters and EBS managed individual credit limits. The EBS limits are local systems that are managed at three Lehman Brothers locations: London (UK), New York (US), Tokyo (JP).

CWS is a Lehman Brothers internal-use only system. We questioned employees in the risk technology department regarding the functionality and underlying data structure of CWS. We also asked them for an overview of the Reuters and EBS systems and how they related to the FX limits we were interested in. Since Reuters and EBS are FX trading applications, we needed to gain an understanding of FX trading. This is explained more in the following Understanding Credit Limits and FX Trading section.

7.1.3 Understanding Credit Limits and FX Trading

To understand credit limits and foreign-exchange trading, we spoke with an analyst from the quantitative risk management department. We learned about the concepts and thoughts behind foreign-exchange trading and how it relates to Lehman Brothers. We also learned about why the process of consolidating limits is important for Lehman Brothers.

7.1.4 Documenting the Current Process

After performing the interviews with previously mentioned Lehman Brothers employees, we documented all knowledge learned from the meetings. We then outlined the sequence of steps that are involved in the current process, and had the users of the process verify those steps to ensure our accuracy. Within the sequence of events, we identified the users and their roles in each step. We developed "use-cases" in order to explain the sequences of the interactions between the users and the systems they used to complete this process. These methodologies showed the functional requirements of the system, the interactions between the systems and the users, and the scenarios developed from each event. Then, from the notes, sequence list, and use-cases, we designed diagrams to display the current process.

7.2 Designing the New Process

Once we developed a clear understanding of the current process, we created a design for our new process. The first step was listing all the functional requirements needed for the new system with respect to the three goals we outlined for our project. We developed use-cases and produced

diagrams showing how the new system will work. We then performed a proof-of-concept analysis to demonstrate the feasibility and flow of the new process and to help us uncover all the possible issues or concerns that could arise. Next we designed the underlying database structure and created an interface prototype to make sure that we met the user's needs. Lastly, we wrote design specifications based on the function requirements, the use-case and the sequence write ups, diagrams, and the scenarios that needed to be addressed.

7.2.1 List Functional Requirements

We listed the functional requirements of the new system to capture the system's intended behaviors and tasks. When developing this we listed the baseline functionality of the system and the additional features that made the new system unique.

7.2.2 Diagram Interactions with New System

We next captured the functional requirements using use-cases. This step was similar to documenting the current process except this was for the new system. Similarly, we also wrote-up the sequence of events for the new system. After the use-case and sequence write-ups for the new process were written, we created diagrams displaying this information graphically.

7.2.3 Proof of Concept

The next step in our methodology was completing a primary proof of concept to demonstrate that the new system design is capable of solving the problems brought to attention. We executed the primary proof of concept, or feasibility of the project, by taking one hundred clients and performing a sample mapping of CWS client names with FX spot credit limits to Reuters and EBS counterparty names. The steps taken in performing this primary proof of concept were as follows:

1. A query of CWS data was run to pull out all counterparties with FX spot credit limits.
2. All the counterparty names (from CWS, Reuters, EBS London, EBS NY, EBS Tokyo) were sorted in alphabetical order.
3. A spreadsheet was created for consolidated viewing with headings:

CWS Name	CWS Line Limit	Reuters Name	Reuters Group Limit	EBS (London) Name	EBS(London) Credit Limit	EBS (NY) Name	EBS(NY) Credit Limit	EBS (Tokyo) Name	EBS(Tokyo) Credit Limit

4. The CWS Name List and CWS Line Limit List were copied and pasted into the appropriate column.
5. Next, based on these counterparty names from CWS, the matching names were found in Reuters, EBS London, EBS NY, and EBS Tokyo. This was done by searching through each

list until the corresponding (possibly) matching name was found. Also, we found during this step, it was beneficial to mark the ones that were matched, i.e. check off on each list from all the systems once the counterparty name was on the new sheet.

6. Once the above steps were completed, the counterparties that are only in CWS, Reuters or EBS were able to be seen as well and were double checked.

After performing the primary proof of concept, we realized that it was possible for Reuters and/or EBS counterparty names that match up with CWS counterparty names to not have internal FX spot credit limits set. Therefore we performed another proof of concept similar to the primary one. During this proof of concept, we performed a sample mapping of the one hundred clients with all CWS client names and Reuters and EBS. We followed the same steps as during the primary proof of concept, but this time we did not filter the CWS data to just counterparties with FX spot credit limits set. From both of these proof of concepts, we had a list of all possible scenarios that could (and did) occur. We then asked the employees for reasons why each of the scenarios would appear and how they would be resolved. Based on our conclusions after the meeting with the users, we made a plan to integrate the business logic of the scenarios and resolutions into the system.

7.2.4 Design the Database Structure to Meet the Needs

After the planning and documenting of the new process was completed, we designed the database structure. This new database structures is shown in an entity relationship diagram. The entity relationship diagram (ERD) visually describes the attributes of entities, or tables, of a database and the relationships between them. The purpose of compiling this diagram was to visually layout the underlying database structure and how it integrated with the existing CWS tables. This ensured that data could be stored and accessed efficiently.

7.2.5 Create Interface Mark-up and Get Feedback

Another aspect of this project was to decide how this data would be displayed. We accomplished this by creating a user interface prototype. Once we had developed this, we presented it to the users to assure it showed all of the information they needed. We requested feedback about usability and features. We wanted to get as much feedback as possible during our project time, to better be able to meet the users' needs.

7.3 Implement the System

Once the design of the system was finished, we began the implementation of the concepts developed. Two steps were involved in integrating the new system. First, we implemented the

database structure, as described in the entity relationship diagram. Second, we implemented the application layer, which itself consisted of two aspects: feed importing and interface applications.

7.3.1 Implement the Database Structure

The database structure stores the data that is imported from the feeds from Reuters and EBS. The development of the database structure included implementing the tables, stored procedures, and views for data access. Lehman Brothers uses Sybase Adaptive Server for their database server. Sybase Adaptive Server is a type of structured query language (SQL) database server. The entity relationship diagram was translated into SQL data description language (DDL) statements, which is used to create the database structure in a database. To access the database tables that were previously implemented, various stored procedures and views were written. Also, to add/delete/modify data in the tables, stored procedures were written. To minimize application-layer dependency, as much business logic as possible was integrated into the database structure. For the same reason, the database layer was implemented before the database structure. This resulted in decreased application-dependent code and implementation detail.

7.3.2 Implement the Application Layer

The next step in implementing the designed system was to implement the application layer. The application layer supports the application and the end-user processes. When completing this layer, we had to implement applications to import the feeds from Reuters and EBS into the database as well as interfaces to access the views/stored procedures in the database. The applications that imported the feeds from Reuters and EBS were Perl scripts with a Sybase DBI adapter. Front-end interfaces for the database were first implemented in Perl, which we later rewritten in Java Server Pages (JSP).

7.4 Produce Documentation

Our last step of the methodology was to produce the documentation for the new system. Since we spent six weeks at the Lehman Brothers London office and two weeks at their New York office, most implementation and production of the system was completed in London. Once we arrived in New York, the details of documenting the new system were completed. This documentation was essential for the success and continuation of our project.

7.5 Test the Implemented System

Our next step of our methodology was to test the implementations within the system. The testing of our system followed the Lehman Brothers protocol of going through a testing phase and then a staging phase. Major changes and bug-fixes occurred in the testing phase, while minor

modifications and robustness testing occurred in the staging phase. Within Lehman, a Q & A (question and answer) team fulfilled the staging phase. Within each phase, testing of the database and application layers was completed.

7.6 Move System into Production Environment

After testing the system thoroughly, it was ready for use in a production environment. Once an application is in the production environment, it is considered to be finished. We followed the internal Lehman Brothers protocol for releasing a product into production. To start this phase, we began to "hand off" the system and release our control of the system. This was more of a "change of ownership" since we are merely temporary employees at Lehman Brothers. The system first had to be approved by managers. Once it was approved, the system was placed and integrated with the production systems. This placement was done during a specified time-frame. To complete specifics of this test, we met with appropriate contacts.

7.7 Conclusions

In conclusion, the methodology we followed during the course of the project was to gather information and implement the system while we were London. Once we were in New York, we finalized documentation and released the system into production. This utilized our time and the resources available at each location. The users and maintainers of the system were in London while the system releasing players were in New York.

8 Results Part 2

The findings and results of our methodology are described in this chapter. From understanding the current process, we produced steps and diagrams to the current process. We then designed and implemented a new process. We then documented the system, and lastly released the system into production.

8.1 Understanding the Current Process and Problem

When understanding the current process, we met with users, understood the systems and sources involved, understood credit limits and foreign-exchange trading, and finally, documented the current process.

8.1.1 Meetings with Users

Through meetings with users we gathered a broad basis for our project as well as many of the fine details. We primarily met with Mahvish Ayoob and Leesan Wong, employees of the risk management department, to obtain information and get referrals to other contacts. The minutes from these meetings can be found in Appendix 2-E: Meeting Minutes. From these meetings we were able to derive information about the current process, ways the new process could be improved and details about the involved systems.

8.1.2 Understanding Systems/Sources Involved

After our meetings with users, we needed to understand the systems and sources involved in the current process. The systems that we investigated were first Credit Work Station (CWS), then Reuters and EBS broker dealer systems.

Credit Work Station is a Lehman Brothers internally developed application that tracks credit risk information for counterparties. For our purposes, we were only interested in the credit lines that were allocated to counterparties. We used these credit limit allocation numbers to compare to those set in Reuters and EBS systems.

The Reuters system used in Lehman is a global broker dealer system that is used internally in Lehman Brothers for keeping track of counterparty foreign-exchange spot and forward credit information. EBS is a similar broker dealer system to Reuters, except it maintains only spot credit limits and different instances of the application are run at the three main Lehman Brothers' offices: Tokyo, New York, and London.

8.1.3 Understanding Credit Limits and FX Trading

To understand credit limits and foreign exchange trading, we met with a credit analyst. The results of these meeting are documented in the background of this report.

8.1.4 Documenting the Current Process

The current, or previous, process was documented through a flow chart diagram. This diagram, shown in Figure 8-A below, visually demonstrates the process flow from beginning to end.

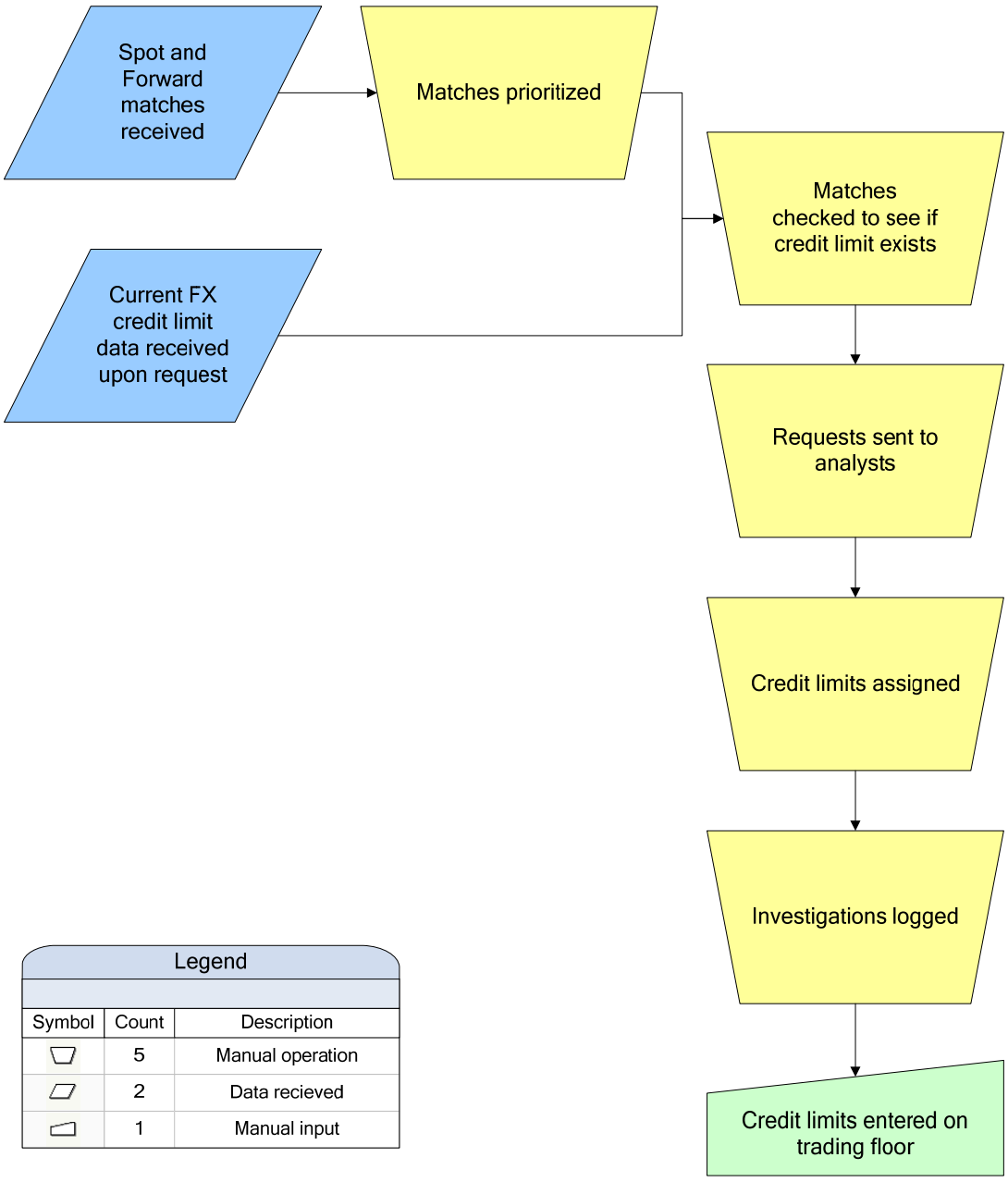


Figure 8-A – Current FX limit consolidation process

The following table describes the steps from Figure 8-A in more detail.

Step	Description
Step 1a: Spot and forward matches received	The first step in the original process involved Reuters and EBS London sending a weekly email to Lehman Brothers' credit risk management department. These emails were sent in various formats and contained spot and forward matches for the previous week.
Step 1b: Matches prioritized	The step that occurs after the spot and forward matches were received was an employee of the credit risk management department would prioritize these matches previously from Reuters and EBS London (step 1a in Figure 8-A). These matches were prioritized based on the number of trade requests.
Step 1 (Alternate): Current FX credit limit data received upon request	Alternatively, a second first step in the former process also involved email being sent containing Reuters and EBS information. However, these emails were sent less frequently and only upon request from the credit risk management department. These emails contained raw data files with current foreign-exchange credit limits stored in Reuters and EBS. The emails from EBS came from three different locations: EBS London, EBS New York, and EBS Tokyo. The emails with the Reuters current FX credit limit data were sent from the Reuters system on the trading floor.
Step 2: Matches checked to see if credit limit exists	The next step consisted of the prioritized spot and forward matches being manually compared to the current foreign-exchange credit limit data that was received (Step 1b) and the limits set in Lehman Brothers internal system, Credit Work Station. This step was completed by an employee of the credit risk management department and was done only when time permits.
Step 3: Requests sent to analysts	Once step 2 is completed, matches remain without credit limits listed in the current FX credit limit data or CWS data. The employee then sent emails requesting the credit limits of the remaining matches. These email requests were sent to the corresponding analysts.
Step 4: Credit limits assigned	Step 4 involved the analysts receiving the request emails and assigning credit limits to the counterparties in the emails. The analysts then set the credit limits based off the creditworthiness and the internal and external credit ratings of the counterparties.
Step 5: Investigations	In step 5, the credit limits from step 4 were then sent back to the

logged	employee in the credit risk management department. The employee then recorded the analysts' investigations and results in a ring binder.
Step 6: Credit limits entered on the trading floor	The final step in the process was when the employee of the credit risk management department went to the trading floor and manually entered the new credit limits. These credit limits were entered into the Reuters and EBS London systems. The timing of when this step was completed depended on the urgency of the credit limits request or the work load of the employee (or both).

8.2 Designing the New Process

The new process utilizing the system that we have implemented has fewer manual steps in the process when compared with the original process. Also, for documenting purposes, the new system was referred to as REBS FX. Below you can see a flow chart similar to that shown in the previous section.

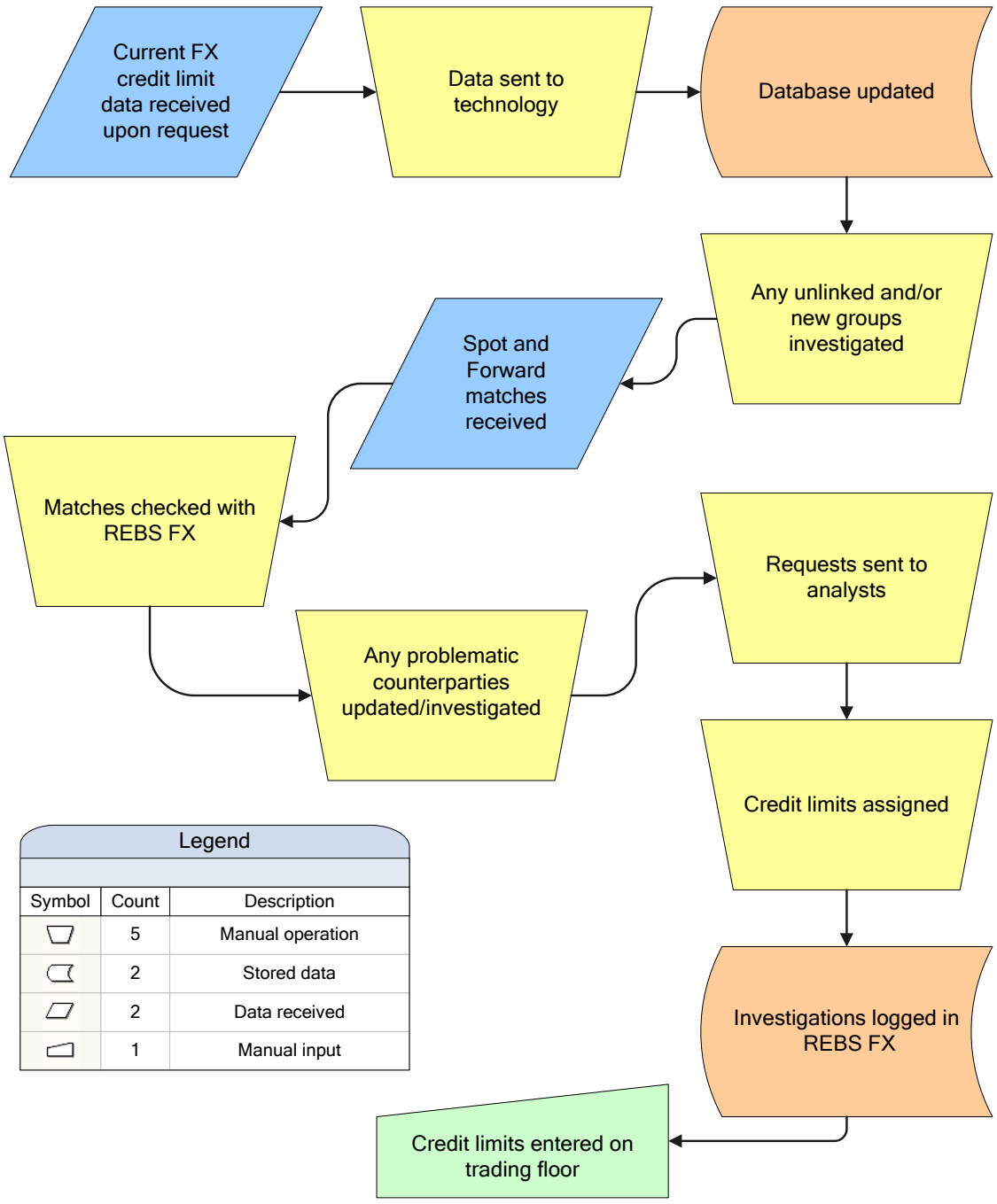


Figure 8-B – Diagram of the new process utilizing implemented system

The following table describes each of the steps from Figure 8-B in more detail.

Step	Description
Step 1: Current FX credit limit data received upon request.	First, similar to the first step in the current process, Reuters and EBS send emails. The emails are sent only upon request from the credit risk management department. In order for the new process to be efficient, these emails should be requested periodically. These emails are raw data files containing the current foreign-exchange credit limits stored in Reuters and EBS. The emails from EBS come from three different locations; EBS London, EBS New York, and EBS Tokyo.
Step 2: Data sent to technology group	The next step involves the employee in the credit risk management department sending the emails from Reuters and EBS locations to a member of the technology team via email.
Step 3: Database updated	Then the member of the technology team updates the database with the new current FX credit limit information from the Reuters and EBS feeds.
Step 4: Any unlinked and/or new groups investigated	Next, any unlinked or new groups that come up in the new system, REBS FX once the database is updated (step 3) are investigated. They are shown on the "Unlinked Entities" report view. Once they are investigated, the unlinked and new groups need to be linked up with the corresponding CWS counterparty legal name. This is done by using REBS FX.
Step 5: Spot and forward matches are received	The fifth step in the new process is the same as one of the first steps in the current process. It involves Reuters and EBS London each sending a weekly email to Lehman Brothers' credit risk management department. These emails are sent in various formats and contain spot and forward matches for the previous week.
Step 6: Matches are checked with REBS FX	During this step, the spot matches received from Reuters and EBS (step 5) are compared with the spot credit limits in REBS FX. This step is done by the employee in the credit risk management department.
Step 7: Any problematic counterparties are updated/investigated	This step occurs if when the previous step was completed, any of the counterparties viewed in REBS FX were problematic or over limit. If this happens the counterparties need to be

	investigated by the analysts and the limits set in either Reuters, EBS, or CWS need to be adjusted.
Step 8: Requests are sent to analysts	Similar to step three in the current process, any remaining matches without spot credit limits listed in REBS FX need to be researched. The employee of the credit risk management department then sends emails requesting the spot credit limits of the remaining matches. These email requests are sent to the corresponding analysts.
Step 9: Credit limits are assigned	The next step is the same as in the current process and involves the analysts receiving the request emails and assigning spot credit limits to the counterparties in the emails. The analysts set the spot credit limits based off the creditworthiness and the internal and external credit ratings of the counterparties.
Step 10: Investigations are logged into REBS FX	After the analysts set new spot credit limits (step 9), they log the investigations and the results using the history feature in the REBS FX.
Step 11: Credit limits are entered on the trading floor	The final step in the new process is when the employee of the credit risk management department then looks up the remaining matches in REBS FX and views the analysts' research results. Then the employee goes to the trading floor and manually enters in the new set spot credit limits. These spot credit limits are entered into the Reuters and EBS London systems. This step should be completed as soon as the analysts have finished the investigations and have set the new spot credit limits.

8.2.1 List Functional Requirements

The functional requirements for the system were derived from the use-cases. These outlined all of the possible ways a user would interact with the system.

8.2.2 Diagram Interactions with the New System

The interactions with the new system were diagrammed in the form of use-case diagrams. These diagrams show how a user would interact with a system. In our case, the following shows how a user interacts with our system and for what purpose.

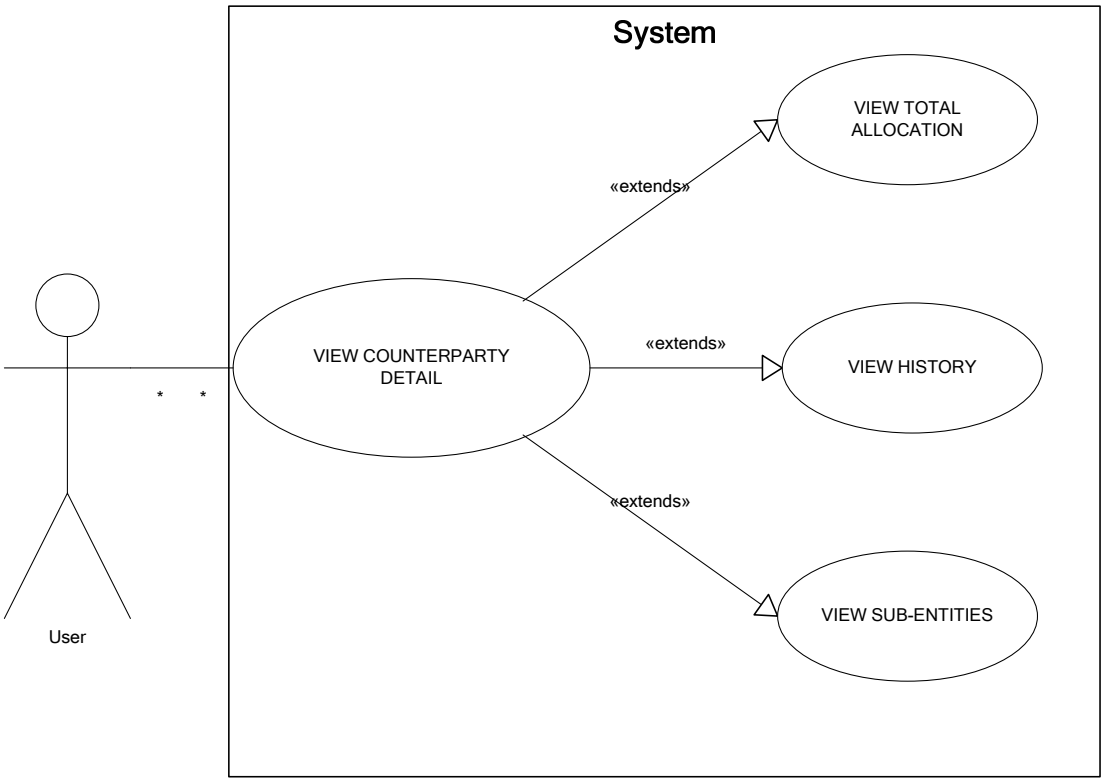


Figure 8-C – Use case diagram showing actors and interactions with system

The above diagram shows how a user can view the details of counterparty. From that detail view, the user can view total credit limit allocation, view all counterparty history details, and view the sub-entities (or branches) of the counterparty.

8.2.3 Proof of Concept

First Proof of Concept

Upon completion of the primary proof of concept, it was proven that the new system was capable of comparing the limits from CWS, EBS London, EBS NY, EBS Tokyo, and Reuters. It also was proven that the system was capable of fulfilling goal one, mapping CWS counterparty legal names to their corresponding Reuters and EBS names. These names are each entered into separate systems, by different people, with no standard naming convention. Given this fact, a manual check is required to match counterparties (based on name). Through the primary proof of concept, it was proven that by using the CWS names with FX spot credit limits as a staring point, the names in Reuters and EBS locations can be matched with those of CWS by searching through each individual sheet for like names.

After completing this primary proof of concept, possible scenarios that could occur were produced. The ideal scenario is that there is a name and a spot limit that exists in Reuters, EBS

London, EBS NY, and EBS Tokyo that corresponds with a name in CWS. However, this is not always the case. The other possible scenarios that could arise were:

1. The counterparty and its spot limit are only listed in CWS.
2. The counterparty and its spot limits are in CWS and EBS Global but not Reuters.
3. The counterparty and its spot limits are in CWS and Reuters but not in EBS Global.
4. The counterparty and its spot limit are only found in CWS and EBS NY.
5. The counterparty and its spot limit are found in CWS, Reuters, EBS London, and EBS Tokyo, but not in EBS NY.
6. The counterparty and its spot limit are only found in CWS, EBS London and EBS Tokyo.
7. The counterparty and its spot limit are found in all sources except EBS London.
8. In CWS the counterparty is listed in two different locations, but in EBS and Reuters it is just listed as one group.

Then, an employee from credit risk management described why each of the scenarios appear and how they should be resolved. The resolutions were as followed:

- If counterparty has a spot limit in CWS and no other sources (scenario 1 above) then it means that there hasn't been a spot limit assigned in Reuters and EBS. This should be flagged to be researched to see if the analysts would like to allocate it in Reuters and EBS.
- If a counterparty has a spot limit in CWS and Reuters but not EBS Global or a spot limit in CWS and EBS Global but not Reuters (scenarios 2 and 3 above) then it more than likely means it hasn't been looked into for that source. This also should be flagged and researched.
- If a counterparty is only in CWS and some EBS location (scenarios 4 and 6 above) then it might be because it not traded globally. This also should be flagged and researched.
- If a counterparty is not found in all the EBS locations (scenarios 5 and 7 above) it may be because no trading is done near the location it is not listed in and then therefore there is no appetite at this location. This also should be flagged and researched.
- In the case where the grouping is different between all the different sources (scenario 8 above), then an investigation needs to be done to make sure all the groups listed are indeed legal entities. This is done by first looking on the client's website to look up subsidiaries and branches. Then second contacting the analyst. This also should be flagged and researched.

Generally, to resolve these scenarios, if a FX spot credit limit is missing in any of the source's information contained in the database; it should be flagged and then adjusted after research. Also, any FX spot credit limit or name changes go through the credit risk management department. When these changes occur, the duplicate FX spot credit limit entry should be deleted.

Second Proof of Concept

Based on our conclusions of the primary proof of concept and meeting with an employee of the credit risk management department, we realized that it was possible for some counterparties in Reuters and EBS to have a corresponding CWS counterparty name and not have an internal FX spot credit limit. We found that fifteen to twenty percent of the counterparty names in Reuters and EBS fit into this category. Therefore, we next completed another proof of concept where we matched one hundred names in Reuters and EBS to CWS names.

This proved to be more efficient and created two more scenarios. The first scenario was that the counterparty name in Reuters and EBS matches up with a CWS counterparty legal name but does not have an FX spot credit limit set in CWS. When this occurs, it needs to be flagged for an analyst investigation to decide whether or not an FX spot limit should be set in CWS. The second scenario was the counterparty name is listed in Reuters and/or EBS but not in CWS. When this occurs, it needs to be flagged and an investigation needs to be done to determine if a new counterparty needs to be created in CWS or the counterparty name needs to be removed from the Reuters and/or EBS systems.

Other Scenarios

As well as the proof of concepts, we produced scenarios that would occur while the system fulfils the other two goals set. The second goal was to create a logging process within the new system to keep track of each client's investigation status. The possible scenarios addressing this goal were:

1. The client has not been previously investigated and an FX spot credit limit is not assigned.
2. The client has been previously investigated and already has an FX spot credit limit assigned.
3. The client has been previously investigated and an FX spot credit limit was not assigned.
4. The analyst is in the middle of investigating the client and the assignment of an FX spot credit limit is pending.
5. The client has been previously investigated and there is a request for an FX spot credit limit increase or decrease.

To fulfill this second goal, a log of the status of each client needs to be kept within the system. Addressing these scenarios, we created three status stages a client can be logged in: never been investigated, pending investigation, or investigated with results. Dividing the above scenarios into the status stages resulted in:

- Scenario 1 above is in the never been investigated stage.
- Scenarios 2 and 3 above are in the investigated stage with results.
- Scenario 4 above is in the pending investigation stage.
- Scenario 5 above is in the investigated stage with results.

Also, when a request for an FX spot credit limit change is made, it progresses to the pending investigation stage until the FX spot credit limit is changed and it returns to the investigated stage with results. Therefore, to resolve scenarios one and four the clients need to be investigated. Scenarios two, three and five above are ideal scenarios; they are resolved and not in need of investigation. When the clients are in any of the above scenarios it needs to be logged.

The third goal was to develop a way to assure that the sub-entities of an entity match up between the two trading applications, Reuters and EBS. The possible scenarios addressing this goal were:

1. There was a sub-entity under a counterparty listed in Reuters, which wasn't listed in EBS.
2. There was a sub-entity under a counterparty listed in EBS, which wasn't listed in Reuters.
3. There was a sub-entity listed under a counterparty in Reuters that was listed under a different counterparty in EBS.
4. There was a sub-entity listed under a counterparty in EBS that was listed under a different counterparty in Reuters.
5. The sub-entities listed under a counterparty are the same for EBS and Reuters.

To resolve the above first four scenarios, investigations need to be done to find out which system has the sub-entities listed correctly. Upon completion of the investigation, the two systems need to be adjusted based on the result. Lastly, scenario five is the ideal scenario.

From our conclusions of both of the proof of concept, all the scenarios, and the meeting with the employee of the credit risk management department, we integrated the business logic of the scenarios and resolutions into the new system.

8.2.4 Design the Database Structure to Meet the Needs

The database was designed to represent all possible data that would be needed to be recorded. The ideas and structure of this database was captured in the entity relationship diagram (ERD), shown in Appendix 2-B: Diagrams. The following tables were created to store the raw data from Reuters and EBS and all other relevant information.

REBS_legal_entities	
This table stores the entity names and limits imported from the feeds (Reuters/EBS). Stored here is also the source from which the limit came from, as well as any other relevant information.	
entityID	Unique identifier for legal entity; generated by database server.
dealing_code	Four-letter unique code for entity; can be null or not exist. Reuters is the only system that currently has entities with dealing codes.
group_name	The name of the legal entity (from Reuters/EBS)
line_amount	The credit line limit amount (from Reuters/EBS)
source	Source from which the limit comes from; will be one of the following: EBSLON, EBSTOK, EBSNYC, or REUTERS
client_code	Foreign key referencing CWS client code. If not linked, it will be -1.
change_date	Date of last change.

REBS_client_history	
This table saves any logged information such as REBS_legal_entity updates/insertions and any user-entered data, such as investigations of the counterparty.	
historyID	Unique identifier for the history entry, generated by the database server.
client_code	Foreign key referencing CWS client code. If not related to a CWS client, it will be -1.
entityID	The entityID foreign key back to REBS_legal_entities. If not related to an entity, it will be -1.
description	Description of the history record.
actiondate	Date the action was done. Generated by the database server.
actioncode	A character code used for querying purposes to identify the action done.
user_code	Windows username of user that inserted the history item.

REBS_subentities	
This table stores the dealing codes and their sub-entity descriptions, imported from Reuters/EBS feeds. These dealing codes are linked back to their parent entities in REBS_entities_subentities.	
subentityID	Unique identifier for sub-entity; generated by database server.
dealing_code	Four-letter unique code for sub-entity/branch; can not be null.
description	The name of the sub-entity/branch from the feed.

REBS_entities_subentities	
This table is the many-to-many relationship of sub-entities to entities. An entity can have more than one sub-entity and a sub-entity can have more than one entity as a parent. This relationship is recorded here.	
entityID	The ID of the parent entity.
subentityID	The ID of the sub-entity.

REBS_legal_entities_log	
This table saves all changes to REBS_legal_entities table. This is done through triggers upon *insertions*, *deletions* and *updates*. If an update is done, a before and after "snapshot" is taken. All history is logged here while only select updates are recorded in REBS_client_history.	
entityID	The ID of the entity that has been changed.
dealing_code	The dealing code of the entity.
legal_name	The legal name of the entity.
line_amount	The credit line limit amount (from Reuters/EBS)
source	Source from which the limit comes from; will be one of the following: EBSLON, EBSTOK, EBSNYC, or REUTERS
client_code	Foreign key referencing CWS client code. If not linked, it will be -1.
change_date	Date of last change.
delete_date	The date the entity was deleted, if deletion. Otherwise, null.

REBS_reports	
This table stores all report information that is used in the interface on the CRM site. The query is saved, sort-able fields, filter-able fields, etc.	
reported	Unique identifier for the report, generated by the database server.
query_string	The SQL query string of the report. This must be a simple select statement. Because this query is filtered by the interface, there can be no `WHERE` or `ORDER BY` clauses in it. If the report must have `WHERE` and/or `ORDER BY` clauses, the query must be implemented in a view and the query string in the report table be a `SELECT * FROM [view_name]` where `view_name` is the name of the view.
description	A description of the report. (Optional)
reportname	A unique name for the report. This is shown on the interface in the reports drop-down list.
author	This is the username of the author of the report.
filter_fields	These are the fields by which the report can be filtered (WHERE X LIKE Y, X is the field; Y is the input from the interface). These must be stored in the field as comma separated fields. For example: `client_code,description,line_amount`.
sort_fields	These are the fields by which the report can be sorted. By default, the report cannot be sorted by any field. These must be stored in the field as comma separated fields, as well. For example: `client_code,action_date`.
check_column	This is a single column which can be checked using a regular expression and be highlighted if the regular expression evaluated to *TRUE*. An example of this would be: `status`.
highlight_regex	The regular expression to be preformed on the check_column field. In the interface, if this evaluates to *TRUE*, that row is highlighted. An example of this would be: `.OVERLIMIT.`
last_run	The date the report was last run.
create_date	The date the report was created. Generated by the database server.

REBS_source_update_history	
This table stores a record of loading data feeds into the system.	
updateID	Unique identifier for the update instance; generated by database server.
filename	The filename of the data feed file.
source	Source from which the feed came from; will be one of the following: EBSLON, EBSTOK, EBSNYC, or REUTERS
file_asof	This is the date that is stored in the file (if from an EBS system). If the data feed was from Reuters, this date cannot be derived from the data file, so it is left as null.
load_date	The date the feed was loaded into the system.

The information stored in the individual tables, as described above, is nearly useless without being related to other tables. To join the data to make relevant views of the information, SQL views were made of the common queries. Described briefly below are the views that were created. The implementation details of these can be found in Appendix 2-C: Database Implementation Details.

REBS_client_history_view
This is a view of the history for a given client/entity. This view merges information of REBS_client_history with counterparty and group names from the CWS clients table and REBS_legal_entities.

REBS_dealing_codes
This is a view of all dealing codes that exist in either (1) REBS_legal_entities or (2) REBS_subentities. This is used to look up the related entity ID for a given dealing code. This is a union of the two tables previously mentioned.

REBS_existinglimits
This is a view of the CWS counterparties with all linked limits shown based on source location. For example, given a CWS client code, the linked source limits for EBS London, EBS Tokyo, EBS New York and Reuters total aggregate limits are shown. Also shown is the total EBS and Reuters limit and the related CWS limit whether or not the counterparty is over their credit limit. The counterparty is over their limit if the EBS/Reuters aggregate sum is greater than the CWS LBI FX SPOT limit. This view is a join of CWS counterparty information, limits, and REBS_legal_entities table information.

REBS_fxspotclients
This is a view of all CWS counterparties with LBI FX SPOT limits in the CWS lines table.

REBS_nonlinkedlimits
This is a view of all entities from the REBS_legal_entities that do not have a link to an existing CWS counterparty. This is usually all entities with a client_code of -1, but also entities that have been linked to a counterparty that does not exist in CWS any more.

REBS_otherlimits
This is a view that is similar to REBS_existinglimits except no CWS counterparty name is shown.

REBS_valid_groups
This is a view of all valid entities that have been linked to a CWS counterparty that has an LBI FX SPOT limit. The inverse of this view would be a table of entities that are either unlinked or linked to a CWS counterparty with no LBI FX SPOT limit.

To change the data in the tables, stored procedures were created. These stored procedures linked the application layer to the raw data in the tables and controlled the flow between the two layers. Below are the stored procedures that were created for the new system, along with their inputs, outputs, and a brief description of the procedure.

REBS_run_report_by_id	
This procedure executes one of the reports from the REBS_reports table. Given a report ID and optionally a WHERE clause string and/or an ORDER BY field(s).	
@reportID	The reportID that corresponds to the REBS_reports table.
@whereString	The WHERE clause that will be inserted. Default is a blank string. An example of this @whereString is: `WHERE line_limit > 0`. Optional.
@orderBy	The field(s) to sort by. If more than one field is specified, they fields should be comma separated. Optional.

REBS_record_subentity	
This procedure is used to create an entry in the REBS_subentities table. If an entry exists based on the dealing code, nothing is done. If there is not an entry in the table, a new one is inserted. The sub-entity ID is returned.	
@entID	The entity ID of the parent.
@dealing_code	The dealing code for the sub-entity.
@description	The description of the sub-entity. This is the name that is associated with the dealing code.

REBS_record_source_update	
This procedure is used to insert a new row into the REBS_source_update_history table. Every time the Reuters/EBS import script is run, this procedure is called with the given parameters.	
@filename	The filename of that was imported.
@source	The source of the feed. Will be one of the following: EBSLON, EBSNYC, EBSTOK, REUTERS
@file_asof	This is the date that the data is up to. This can be null, for example, in the Reuters case where no "as-of" date is saved in the feed data. Optional.

REBS_record_entity	
This procedure inserts or updates the REBS_legal_entities table given the parameterized data. If the entity is not in the table, it is inserted with the given data. If the entity already exists, it is checked for an update and updated if the information has changed (ie. line_amount). This procedure is called for each entity that is read in from the data feeds.	
@source	This is the source for the entity.
@entity_name	This is the name of the entity.
@line_amount	This is the credit limit amount that is from the feed data.
@dealing_code	This field is the four-letter dealing code. Optional.
@client_code	This field is the CWS client_code that the entity relates to. By default, this value is -1, or not linked to CWS counterparty.

REBS_record_client_history	
This procedure is for making entries into the REBS_client_history table. This procedure is called from either the table triggers (selectively) or from the interface when keeping a log of manual activity for an entity/counterparty.	
@entityID	The entity ID that the history entry relates to. Default is -1.
@action_code	The action code that is associated with the history.
@description	A description of the action that was taken or a description of the history entry that was made.
@client_code	The CWS client_code that the history entry relates to.
@user_code	The Windows user name that made the entry. Default is 'sys' if the system performed the entry.

REBS_link_entity	
This procedure links an entity to a CWS counterparty. This is called from the interface when linking/unlinking counterparties to entities. This procedure calls REBS_record_client_history as well as updating the REBS_legal_entities table.	
@entityID	The entity ID that relates to the entity.
@client_code	The new CWS client_code that the entity should be linked to.
@user_code	The Windows username of the user that performed the linking.

REBS_entity_subentities	
This procedure returns a table of the sub-entities that are related to an entity. This procedure joins REBS_legal_entities with REBS_subentities using REBS_entities_subentities.	
@entityID	The entity to base the query off.

REBS_entity_details	
This procedure gets the details for an entity. This runs a query that joins REBS_legal_entities and the CWS clients and lines tables.	
@eID	The entity ID of which to get the details for.

REBS_cws_client_details	
This procedure gets the details for a CWS counterparty information includes LBI FX SPOT credit limit and other relevant counterparty information. This procedure joins CWS clients and lines tables.	
@cc	The client_code of the CWS counterparty to get the details for.

REBS_client_subentities	
This procedure gets the sub-entities that are related to a CWS client. Returned is a table of dealing codes, sub-entity names, and Reuters, EBS London, EBS New York, and EBS Tokyo counts of where the dealing code is linked to.	
@client_code	The CWS client_code to get the sub-entities for.

REBS_assoc_entities	
This procedure gets the entities that are associated to it, based on CWS client_code. This will return a table of entities with the same client code as the entity ID parameter.	
@eID	The entity ID of the entity to get related entities for.

8.2.5 Create Interface Mock-up and Get Feedback

An interface prototype was initially done in basic HTML. This can be seen below in Figure 8-D.

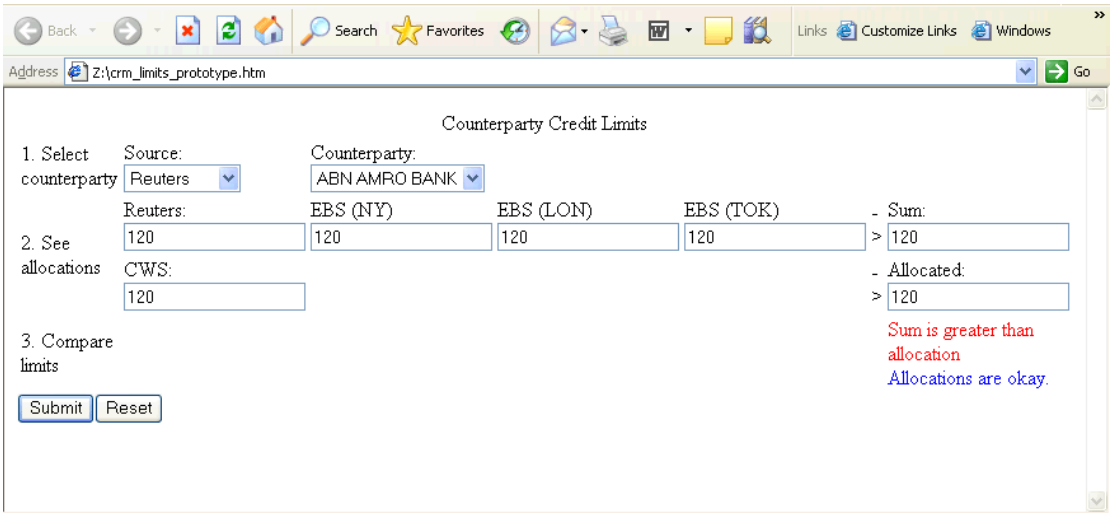


Figure 8-D – Initial interface prototype

8.3 Implement the System

We divided the implementation of the system into two sections: implementing the database structure and implementing the application layer(s). The implementations of these sections were done with Lehman Brothers database and application infrastructures in mind. The details of these implementations can be found in Appendix 2-D: Application Layer Implementation Details.

8.3.1 Implement the Database Structure

Lehman Brothers technology departments generally use Sybase SQL servers. These servers use a variation of Transact-SQL as their language for describing the tables in the relational database. The implementation of the database was done by translating the entity relationship diagram from the previous section into data descriptor language, a computer-readable version of the diagram. The data descriptor language (DDL) was written using Transact-SQL. These DDL statements can be found in the Appendix 2-C: Database Implementation Details.

8.3.2 Implement the Application Layer

The application layer(s) were divided in two basic sections: data importing and interface. The data importing application handled the parsing and importing of raw data files from Reuters and EBS and placing the data into the database structure, previously implemented. The interface was implemented to interface on top of the database structure and to abstract the details of the database.

The importing applications were written in Perl as UNIX shell scripts. There is one script for parsing and importing the data from Reuters and one for the data from EBS. These two scripts call functions in a Perl include file, `dbaccess.pl`. These scripts were later modified to match the structure and style of existing Lehman Brothers automated data feed importing scripts. The details of the original implementation of these scripts can be found in Appendix 2-D: Application Layer Implementation Details.

The second application that was implemented was the interface. The interface was originally implemented as a common gateway interface script written in Perl. This was later rewritten in Java Server Pages. The Structure and flow of the JSP pages is shown in Figure 8-E.

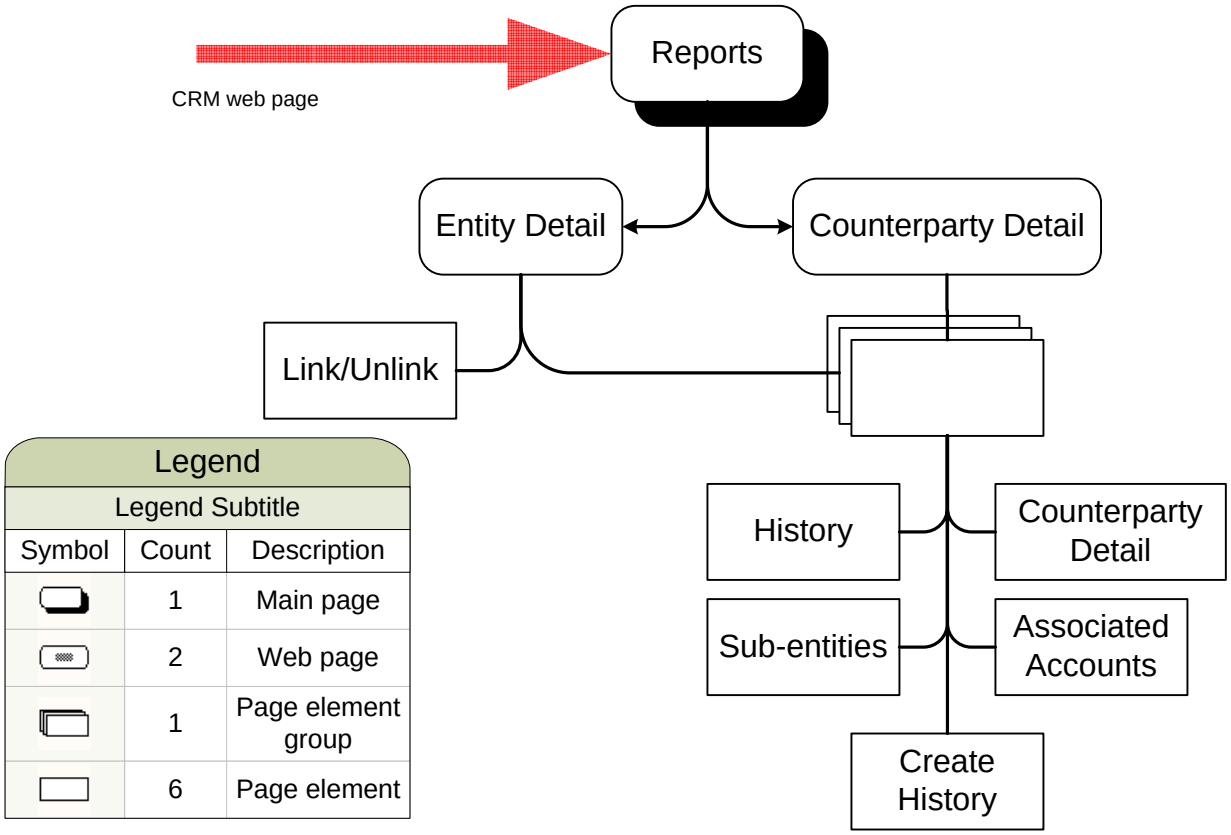


Figure 8-E – Page structure of the JSP web page implementation

The above diagram shows the page flow. The user first views the reports page. From the reports page, the details of entities or counterparties can be viewed. The entity detail page has everything the counterparty detail page has as well as a link/unlink page element.

A screenshot of the reports page with the default report selected is shown in Figure 8-F. The default report is determined by the lowest report ID, stored in the database.

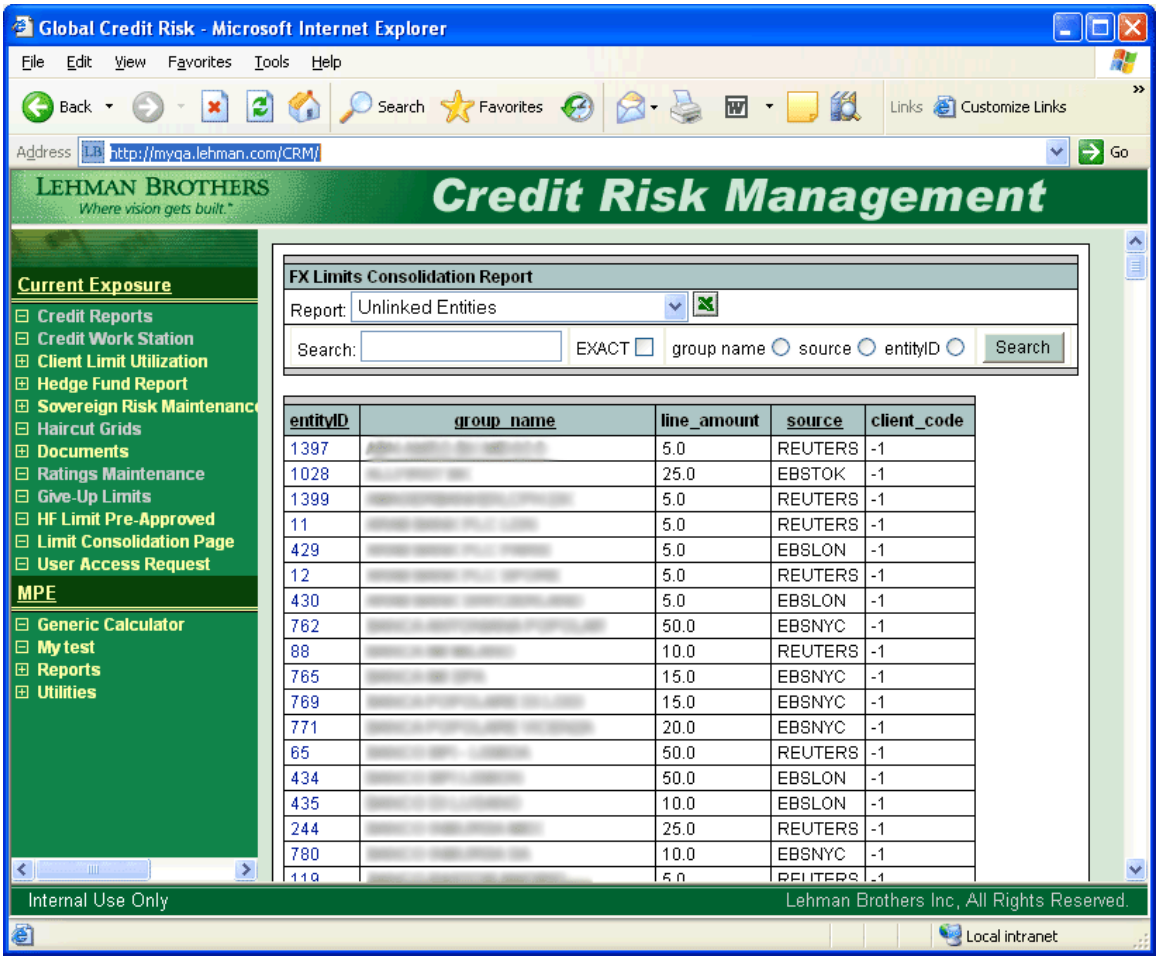


Figure 8-F – Interface screenshot: reports

The entity detail and counterparty detail pages are "pop-up" windows that open a new web browser window when an entity ID or CWS client code is clicked. The entity detail page can be seen in Figure 8-G. From this page, the entity can be linked or unlinked to a CWS counterparty. The functionality of looking up a CWS counterparty mimics that of the Credit Work Station system. A counterparty name is searched by typing the beginning characters of a counterparty name and all possible matching counterparties are shown in a drop-down list.

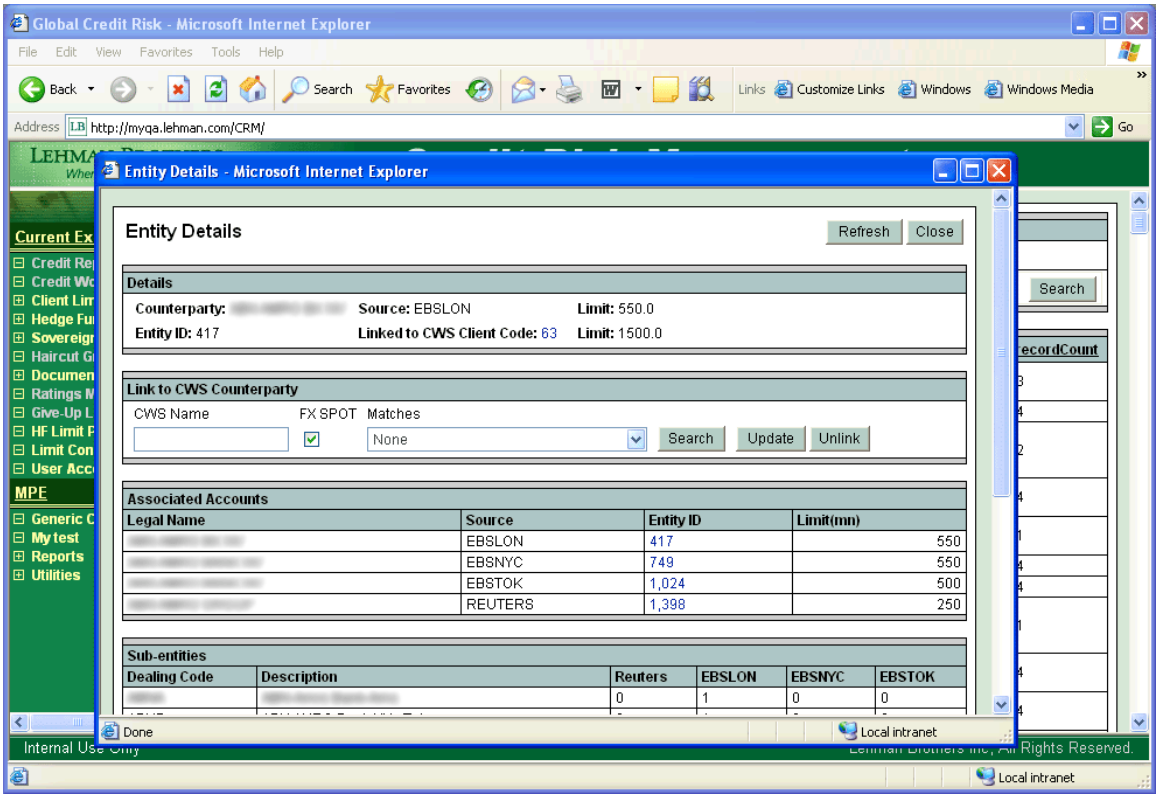


Figure 8-G – Entity detail pop-up page

The client details page can be seen in Figure 8-H. From the client details page, the associated accounts, sub-entities, and client-related history can be viewed. The client-related history shows all history items that are related to the client code of the counterparty being viewed.

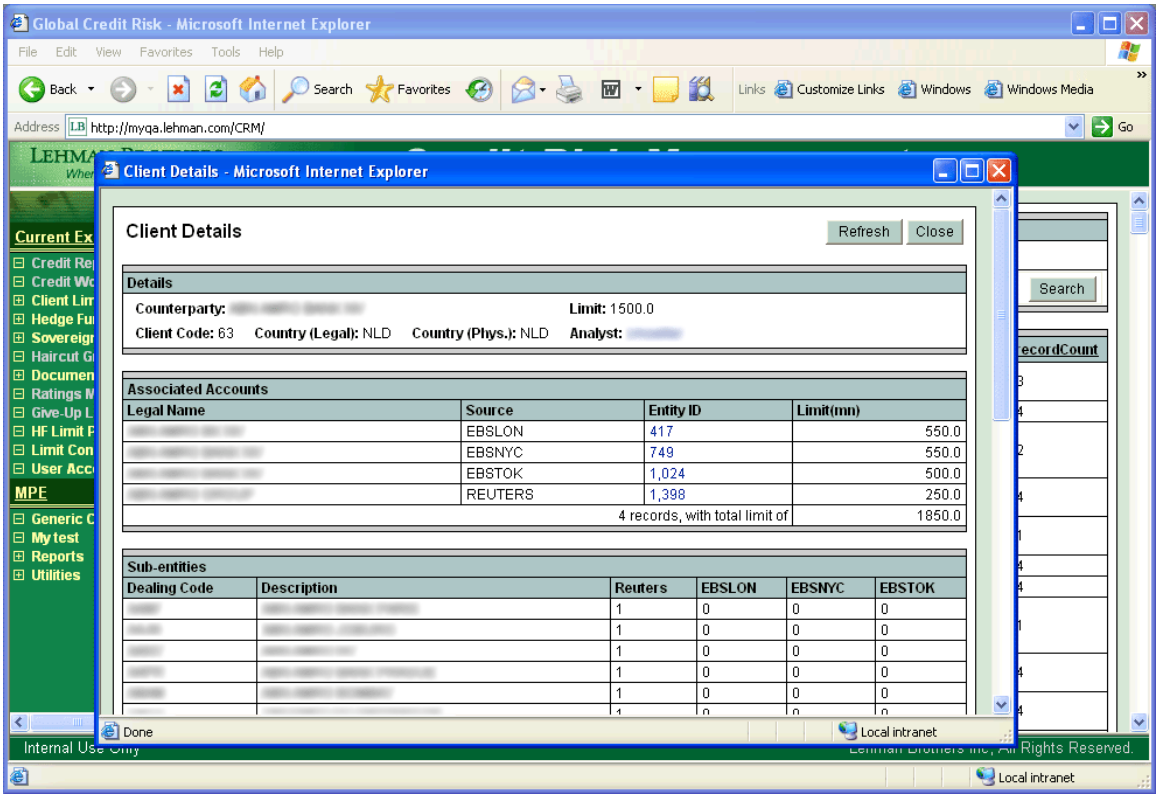


Figure 8-H – Client detail pop-up page

8.4 Produce Documentation

Once the database and application implementations were completed we finished the documentation. The primary documentation deliverable from this project is this report. This report reviews all aspects of the system. This documentation will be used by all users and developers of the system. This document also covers the project process from start to end.

8.5 Test the Implemented System

After the system was implemented, it needed to be tested. Testing of the system was initially done based on the scenarios and use cases that were developed. The next and final step in the testing was user acceptance testing, or UAT. This testing is completed by the users of the system, namely Leesan Wong, in our case.

8.6 Move System into Production Environment

The final step of our methodology was moving the system into production. When we moved the system from testing to production, meetings with database and CRM web site employees were done. These meetings resulted in Lehman Brothers employees being knowledgeable of the inner-workings of our system, from database to application details.

9 Conclusions Part 2

In conclusion ,the process of comparing foreign-exchange spot limits across multiple broker dealer systems within Lehman Brothers has greatly improved through the development and implementation of the system completed in this project. Our conclusions can be divided into three sections based on the different types of goals set for the project: user, technical and informational goals. In addition, the implemented system saves time and makes the process steps more efficient.

9.1 User-based Conclusions

The implemented system, based on the set of user goals, achieved the following:

- **A consolidated view of credit limits**
- **Efficient mapping of counterparties**
- **A log of all counterparty activity**
- **A method for comparing sub-entities from Reuters and EBS.**

First, foreign-exchange spot credit limit representation is now shown in one consolidated global view across different broker systems. This enables a user to easily compare counterparty limits and evaluate limit allocations.

Second, the mapping of Reuters and EBS counterparties to CWS counterparties is done efficiently. This allows for an effective prerequisite for comparing the FX limits. Without this step being efficient, time would be wasted repeating work.

Third, the ability to log counterparty investigations and other history is fulfilled by the new system. Keeping a log of information eliminates the need for a paper trail log which existed in the former process. This need for a paper log was eliminated by the new system. For this reason, the efficiency when keeping a log is greatly increased.

Lastly, the ability to view and compare sub-entities of Reuters and EBS counterparties is possible. Previously, there was no automated or efficient means of comparing branch groupings. Now, with the new system, these sub-entities can be easily compared.

9.2 Technically-based Conclusions

The implemented system met the technical goals that were set. The system is integrated into an existing credit risk infrastructure, allowing future modification of the system to be easily done by an employee who is knowledgeable of the system. The system is integrated into the Credit Risk Management (CRM) intranet site. The interface is implemented using similar techniques as those of similar CRM pages. When implementing the interface, a Lehman Brothers employee, Bappa Roy, worked closely with the project team to assist in the integration. Because of this, modifications and

further implementation of the system can be done without as much of a learning curve for the system.

9.3 Information-based Conclusions

One goal we had was to learn about foreign-exchange limits and how they were set. In conclusion of this goal, we learned the financial background and reasoning for this project. With thorough knowledge of foreign-exchange trading, risk and limits, we were able to more accurately design the system to not only meet the needs of the users, but to exceed them.

9.4 Generalized Conclusions

The steps in the process of comparing foreign-exchange spot credit limits for a subset of counterparties took around three hours to complete. With the new process, using the developed system, the same process decreases the amount of time required by approximately 28 percent when compared with that of the former process. An estimate of the amount of time for each step can be seen in the table below. The estimates for the amount of time for each step are for a batch of 7 to 10 counterparties.

Former Process		New Process	
Step	Time (h:mm)	Step	Time (h:mm)
1 Spot and forward matches received	0:00	1 Current FX credit limit data received upon request.	0:00
2 Matches prioritized	0:05	2 Data sent to technology group	0:05
3 Current FX credit limit data received	0:30	3 Database updated	0:00
4 Matches checked for existing credit limit	0:30	4 Any unlinked and/or new groups investigated	0:10
5 Requests sent to analysts	0:15	5 Spot and forward matches are received	0:00
6 Credit limits assigned	0:45	6 Matches are checked with REBS FX	0:10
7 Investigations logged	0:15	7 Any problematic counterparties are updated/investigated	0:15
8 Credit limits entered on trading floor	0:20	8 Requests are sent to analysts	0:05
		9 Credit limits are assigned	0:45
		10 Investigations are logged into REBS FX	0:05
		11 Credit limits are entered on the trading floor	0:20
Total manual time:	2:40	Total manual time:	1:55
		Time difference:	0:45
		Percent time decrease:	28%
All times are estimates for a group of approximately 7-10 counterparties.			

Table 9-A – Quantitative advantages of the new system

Many of the steps for the former process are repeated in the new process. The addition of automated steps into the new process saves time and makes the new process more efficient. Even though there are more steps in the new process, the new system allows for many time-saving advantages. The steps in the new system are more automated and more strictly logged. In the prior

process, steps were logged through a paper trail In contrast, all changes/information are logged electronically, including information that was not previously recorded.

10 Recommendations Part 2

After completing this foreign-exchange spot credit limit consolidation project, we have some recommendations. Although we met all of our set goals and successfully implemented the new system, there are some future suggestions that we would like to make both within the scope of our project and outside. Completion of these recommendations would help to maintain and improve this new system and process. We divided our recommendations into two categories: post project and long term. Post project recommendations are suggestions for action immediately after the completion of our project. Long term recommendations are general suggestions for action any time after the completion of our project.

10.1 Post Project Recommendations

We suggest post project recommendations have action taken immediately upon completion of this project. Our post project recommendations are:

1. **For the new system to be periodically updated with the Reuters and EBS feeds.**
2. **For the new system to be maintained.**

Our first post project recommendation involves the current foreign exchange credit limit feeds that are received from Reuters and EBS. Currently, these feeds are not received on a consistent nor periodic basis, leaving the data which the spot and forward matches are compared to out of date. For this reason, we suggest these feeds to be requested on a periodic basis by the credit risk management department. Having these feeds received periodically will assure that the data consistently contains current foreign-exchange credit limit data.

Second, we recommend the system is maintained. Specifically, we recommend:

1. **The remaining unlinked Reuters and EBS entities to be linked with their corresponding CWS counterparty names.**
2. **Any counterparties with total Reuters and EBS FX spot credit limits that exceed the CWS set limit to be investigated.**
3. **The history action codes in the new system to be created and standardized.**

At the time of the publishing of this report, there were approximately 500 entities listed in Reuters and EBS that are not linked to a counterparty name in CWS. In order to clean-up the data contained in the new system and assure it is current, these unlinked entities need to be linked. Therefore, we suggest an employee in the Lehman Brothers credit risk management department researches these unlinked counterparties and links them to CWS counterparty names.

Also, when comparing FX spot credit limits, some counterparties have total Reuters and EBS limits which are over the limit appetite that is set in CWS. A counterparty is considered to be

over limit when its total FX spot credit limit in Reuters and EBS exceeds the limit in CWS that was set by the analyst. We recommend these counterparties to be investigated by the credit risk analysts. Once these have been investigated, either adjusts to the limits set in Reuters, EBS or CWS should be made based on the results of that investigation.

Lastly, we recommend that the action codes for client/entity history entries to be standardized. This is a feature that allows the history of a counterparty to be entered into a history log. We suggest the credit risk management department decide on the possible history action codes and standardize them. This will ensure the history of each counterparty is well organized and all logged using the same standard. Having these history entries organized well will allow for future database queries to be made efficiently.

10.2 Long Term Recommendations

We consider our second grouping of recommendations to be long term. Long term recommendations will more generally improve the new system and process at some point after the release of the system. In the long term we recommend:

1. **The feeds containing current FX credit limits to be received from Reuters and EBS automatically.**
2. **Lehman Brothers' CRM implementation standards to be made readily available.**
3. **The new system to be used and maintained globally.**
4. **The new system to be extended to include other product lines and broker systems.**
5. **Lehman Brothers continues to take on WPI projects to aid credit reporting in London.**

Currently, the current FX credit limit data is obtained by the credit risk management department through email. This step in the process is very manual and requires employee time. In order to improve this, we recommend Lehman Brothers request feeds from Reuters and EBS to be sent automatically, thus guaranteeing the information reaches the database efficiently. Once this step is automated, this will reduce the total process time as well as reduce the possibility for human-error.

When we started this project, the CRM implementation standards were not readily available. In hindsight, we suggest that Lehman Brothers releases architecture and framework documentation for the CRM website and makes it easily accessible to all Lehman office locations. This will help to have a standard set of architecture which will be used throughout Lehman Brothers.

This new system was built to aid London's credit risk management department with comparing FX spot credit limits. However, the system also contains the FX spot credit limits

maintained in the Lehman Brothers' New York and Tokyo offices. We recommend this new system to be used and maintained globally, overall improving Lehman Brothers FX trading.

The new system contains the FX spot credit limits. We suggest it to be extended to include other product lines and broker service systems. This will allow the information about different products available in one consolidated view. For example, we suggest the system is extended to include FX forward credit limits. This information also comes from Reuters, making it easy to extend the system to include this information.

Finally, we recommend Lehman Brothers takes on or sponsors future WPI Major Qualifying Projects (MQPs) to benefit the credit risk management department. We based this recommendation on the results and deliverables of our project. We also suggest the students of future projects understand our approach and methodology and use it as a tool for the start of their work.

Appendix A: Glossary of Terms

Term	Definition/Explanation
ARF	Asset Risk Factor
Beta	A risk metric that measures the systematic risk of a single instrument or entire portfolio.
CDF	Counterpart Default Factor
CLS	Continuous Linked Settlement (CLS) allows foreign exchange transactions to be settled within the same day.
CRM	CRM (Credit Risk Management) is a web portal that is used internally within Lehman Brothers for "credit risk management."
CWS	CWS (Credit Work Station) is a part of the Lehman Brothers Credit Risk Management (CRM) intranet site.
dmsExtract	Data table containing cross currency swap data, found in the NY_GCREDIT_DEV2 database
EBS	Electronic foreign exchange spot dealing system; Electronic broker system. (See www.ebs.com)
ELP	Expected Loss Potential (loss measurement based on stressing down positions using risk factors derived from previous EMG crisis)
Entity	Within the scope of this report, an entity is used to describe a counterparty that is in one of the Reuters or EBS systems.
ERD	ERD (entity relationship diagram) is a database model that describes the attributes of entities and the relationships among them (Computer Desktop Encyclopedia, 2005).
ESM	Standard Product and Pricing system in Lehman
Forward	A trade that is settled on a future date.
FX	FX (foreign exchange) is a transaction of international monetary business, as between governments or businesses of different countries (Houghton Mifflin Company, 2005).
Gamma	The Greek factor sensitivities measuring a portfolio's second order (quadratic) sensitivity to the value of an underlier. The rate of change of an option's delta with respect to underlying price. The second derivative of option value with respect to underlying price. Defined as second partial derivative.
GPP	Global Pricing and Product

Greeks	A set of factor sensitivities used for measuring risk exposures related to options or other derivatives. Commonly used to indicate an options value and how this value will change as market conditions change. Defined as first partial derivative.
HJM	Data source system used in data analysis of Part 1 of the report. Includes swap related data for emerging market countries
LOD	Loss on Default (more extreme measure of potential loss a kin to a worst case scenario loss)
Matches	Potential trades.
MUREX	Data source for FX data
Reuters	Electronic foreign exchange spot and forward trading global system.
Spot trade	A trade for spot settlement.
SUMMIT	Data source system used in data analysis of Part 1 of the report. Includes swap related data for emerging market countries
Time Series	A series of observations made over a period of time.
UAT	User acceptance testing.
Underlying/ Underlier	The instrument which the option is based or written on. This can be any tradable instrument which has a defined market price. Common examples include stocks, commodities and cash index.
VaR	Value-at-risk, probability of market risk
Volatility	A metric of variability in a stochastic process. The degree to which the underlying price tends to fluctuate over time. Historical volatility can be calculated by looking at price fluctuations over a specific period in the past. Implied volatility can be implied from option prices observed in the market place. This is achieved by using the Black Scholes Equation, or one of its derivatives to calculate an option volatility which gives the current market option price. Historical and implied volatility can be used to estimate the price of OTC options.

Appendix 1-B: Data Analysis Results

The following reviews the results of the data analysis.

		prodtype is SWAP		Total Items = 1086						prodtype is Null	Total Items = 100		
	use for segregation	*Use for valuation*	*comment*	example 1 (IR Leg 1)	Example 1 (IR Leg 2)	Additional Comments / Questions	*use for segregation*	*Use for valuation*	*comment*	example 1 (IR Leg 1)	Example 1 (IR Leg 2)	Additional Comments / Questions
rowNumber	no	no		346739986	346739985		no	no		347149907	347149908	
host	no	no		DMS	DMS		no	no		DMS	DMS	
site	yes	no	NY(1020), LDN(66)	NY	NY		no	no	always TKO	TKO	TKO	
sys	no	no	total = 1086	HJM	HJM		no	no	always HJM	HJM	HJM	
legid	yes	no		1002525L_I2	1002525L_I1		yes	no		986037L_I1	986037L_I2	
focusid	yes	no		1002525L	1002525L		yes	no		986037L	986037L	
size	maybe	yes		50000000.0	50000000.0		maybe	yes	0.0(66)	10000000000	10000000000	
initNotl	maybe	yes	is = zero(106) ! = zero(980)	50000000.0	50000000.0		maybe	yes	0.0(76)	0.0	0.0	
eff	maybe	maybe		20050822	20050822		maybe	maybe		20060720	20060720	
mat	maybe	maybe	some mat dates have passed	20060822	20060822		maybe	maybe		20150720	20150720	
tdate	maybe	maybe		20050809	20050809		maybe	maybe		20060720	20060720	
optPorS	yes	maybe	[NULL](10) Blank Cell(32) P(888) S(156)	P	P	should not be data for SWAPs in this field	yes	maybe	[NULL](12) Blank Cell(64) P(16) S(8)	P	P	
subType	no	no	always [NULL]	[NULL]	[NULL]		no	no	always [NULL]	[NULL]	[NULL]	
amort_ind	no	no	always [NULL]	[NULL]	[NULL]		no	no	always [NULL]	[NULL]	[NULL]	
basis	no	yes	NULL(27) 30/360(574) A/360(6) ACT/360(396) ACT/365(33) ACT/ACT(50)	30/360	ACT/360		no	yes	0(50) A/360(50)	A/360	0	
description	no	no	NULL(10) 0(32) Blank Cell(1044)	Blank	Blank		no	no	always Blank	Blank	Blank	
exerciseType	maybe	no	NULL(10) Blank Cell(32) A(48) B(928) E(68)	B	B	need further explanation of this field	maybe	no	0(76) B(24)	B	B	
first_exe	no	no	dates vary 1 = 99991231??	20050811	20050811		no	no	dates vary	20060720	20060720	
fixFloat	no	no	always [NULL]	[NULL]	[NULL]		no	no	always [NULL]	[NULL]	[NULL]	
intIndex	no	no	always [NULL]	[NULL]	[NULL]		no	no	always [NULL]	[NULL]	[NULL]	
intSpread	no	no	always 0.0	0.0	0.0		no	no	always 0.0	0.0	0.0	
nextResetDate	no	no	always [NULL]	[NULL]	[NULL]		no	no	always [NULL]	[NULL]	[NULL]	
optExpDate	no	yes	some are already expired	20100211	20100211		no	yes	dates vary, majority is [NULL](70)	[NULL]	[NULL]	
optionType	yes	maybe	[NULL](10) Blank Cell(32) Call(996) Put(48)	Call	Call	should not be data for SWAPs in this field	yes	maybe	0(76) Call(24)	Call	Call	
payccy	maybe	yes	USD(1020) EUR(66)	USD	USD		maybe	yes	HKD(76) KRW(24)	KRW	KRW	

Table B - HJM Swap Analysis

	use for segregation	*Use for valuation*	*comment*	**example 1 (IR Leg 1)**	**Example 1 (IR Leg 2)**	**Example 2 (xCCY Leg 1)**	**Example 2 (xCCY Leg 2)**
rowNumber	*no*	*no*		342627281	342627282	342643341	342643342
host	*no*	*no*		DMS	DMS	DMS	DMS
site	*yes*	*no*	*LDN/NY (1:2)*	LDN	LDN	LDN	LDN
sys	*yes*	*no*	*total = 322,288*	SUMM	SUMM	SUMM	SUMM
legid	*yes*	*no*		100404L.63780	100404L.63781	180184E.186444	180184E.186445
focusid	*yes*	*no*		100404L	100404L	180184E	180184E
size	*no*	*yes*	*size != initNotl 5867 times; size =zero- 56 times;*	-200000000.0	200000000.0	-27000000.0	6775000.0
initNotl	*no*	*yes*		-200000000.0	200000000.0	-27000000.0	6775000.0
eff	*no*	*maybe*		19980421	19980421	20010706	20010706
mat	*no*	*maybe*	*some mat dates have passed*	20080421	20080421	20060706	20060706
tdate	*no*	*maybe*		19980421	19980421	20010704	20010704
optPorS	*no*	*no*	*always null*	[NULL]	[NULL]	[NULL]	[NULL]
subType	*no*	*no*	*always null*	[NULL]	[NULL]	[NULL]	[NULL]
amort_ind	*no*	*no*	*mostly null, except for a few; see values*	[NULL]	[NULL]	[NULL]	[NULL]
basis	*no*	*yes*		A365F	A365F	ACT	A360
description	*maybe*	*no*	*30,812 not null; looks like issuer ID when filled in*	[NULL]	[NULL]	[NULL]	[NULL]
exerciseType	*no*	*no*	*always null*	[NULL]	[NULL]	[NULL]	[NULL]
first_exe	*no*	*no*	*either 99991231 or null*	[NULL]	99991231	[NULL]	[NULL]
fixFloat	*yes*	*yes*		FLO	FIX	FIX	FLO
intIndex	*no*	*no*	*generic term for index - not reliable*	ZARBA	FIXED	FIXED	LIBOR
intSpread	*no*	*yes*	*if fixFloat = 'FIX', contains 0.0, if fixFloat = 'FLO' contains spread over base*	0.0	0.0	0.0	-0.04
nextResetDate	*no*	*yes*	*if fixFloat = 'FIX', contains 99991231; if fixFloat = 'FLO', contains next reset date of index value*	20060123	99991231	99991231	20060104

Table C – SUMMIT Swap Analysis

		Total = 12,716		HJM prodtype = SWAPTION Analysis				Total = 44245			SUMM prodtype = SWAPTION Analysis		
	use for segregation	Use for valuation	comment	example 1 (IR Leg 1)	Example 1 (IR Leg 2)	Additional Comments / Questions	use for segregation	Use for valuation	comment	example 1 (IR Leg 1)	Example 1 (IR Leg 2)	Additional Comments / Questions
rowNumber	no	no		350373118	350373119		no	no		354623302	354623303	
host	no	no	DMS	DMS	DMS		no	no	DMS	DMS	DMS	
site	yes	no	LDN(854) NY(11862)	LDN	LDN		yes	no	LDN(17322) NY(26923)	NY	NY	
sys	yes	no	HJM	HJM	HJM		yes	no	SUMM	SUMM	SUMM	
legid	yes	no		AR2104971_l1	AR2104971_l2		yes	no		1034301L.763N6T	1034301L.763N6V	
focusid	yes	no		459908L	459908L		yes	no		1034301L	1034301L	
size	maybe	yes	vary from 9000 to 10,000,000,000	25000000	25000000		no	yes	vary	-2000000000	2000000000	
initNotl	maybe	yes	0.0(242)	25000000	25000000		maybe	yes	vary	-2000000000	2000000000	
eff	no	maybe	majority 20051123(248)	20050223	20050223		no	maybe	vary	20081001	20081001	
mat	no	maybe	some dates have passed	20350223	20350223		no	maybe	some dates have passed	20101001	20101001	
tdate	no	maybe		20050218	20050218		no	maybe		20050928	20050928	
optPorS	maybe	no	P(7854) S(4862)	P	P		maybe	no	P(23854) S(20391)	P	P	
subType	no	no	always [NULL]	[NULL]	[NULL]		no	no	always [NULL]	[NULL]	[NULL]	
amort_ind	no	no	always [NULL]	[NULL]	[NULL]		no	no	[NULL](44239) A(6)	[NULL]	[NULL]	
basis	maybe	yes	majority ACT/360(6492) [NULL](252)	[NULL]	30/360		maybe	yes	majority A360(21423) 30/360(17378)	A365F	A365F	
description	maybe	no	[NULL](8987) Blank(2999)	Blank	Blank		maybe	no	vary but most are [NULL](40164)	[NULL]	[NULL]	
exerciseType	no	no	A(976) B(10774) E(966)	B	B		no	no	A(270) B(886) E(43089)	E	E	
first_exe	maybe	no	dates vary	20150223	20150223		no	no	[NULL](28735) 99991231(15510)	[NULL]	[NULL]	
fixFloat	no	no	always [NULL]	[NULL]	[NULL]		maybe	no	FIX(22092) FLO(22153)	FIX	FLO	
intIndex	no	no	always [NULL]	[NULL]	[NULL]		maybe	maybe	majority FIXED(22092) LIBOR(14747) EIBOR(7064)	FIXED	BKBIL	
intSpread	no	no	always 0.0	0.0	0.0		no	maybe	majority 0.0(43441) the rest vary	0.0	0.0	
nextResetDate	no	no	always [NULL]	[NULL]	[NULL]		no	no	majority 99991231(42188)	99991231	99991231	
optExpDate	no	no	dates vary with [NULL](2)	20300223	20300223		no	no	dates vary	20080929	20080929	
optionType	maybe	yes	Call(10604) Put(2112)	Call	Call		no	no	[NULL](28735) Blank Cell(10384) NONSTANDARD(5126)	[NULL]	[NULL]	
payccy	maybe	yes	USD(12010) EUR(704) JPY(2)	EUR	EUR		maybe	yes	majority USD(21018) EUR(14385) JPY(7106) GBP (1566)	TWD	TWD	
payfrq	maybe	yes	M(3154) Q(5717) S(3569) Y(264) Z(12)	Y	S		maybe	yes	A(6896) M(225) Q(9858) S(27261) Z(5)	Q	Q	
rate_float_leg	no	no	always 0.0	0.0	0.0		no	maybe	majority -1.0(44068) the rest vary	-1	-1	

Table D – HJM SWAPTION Analysis

Appendix 1-C: Country Risk FX Product Comparison Table

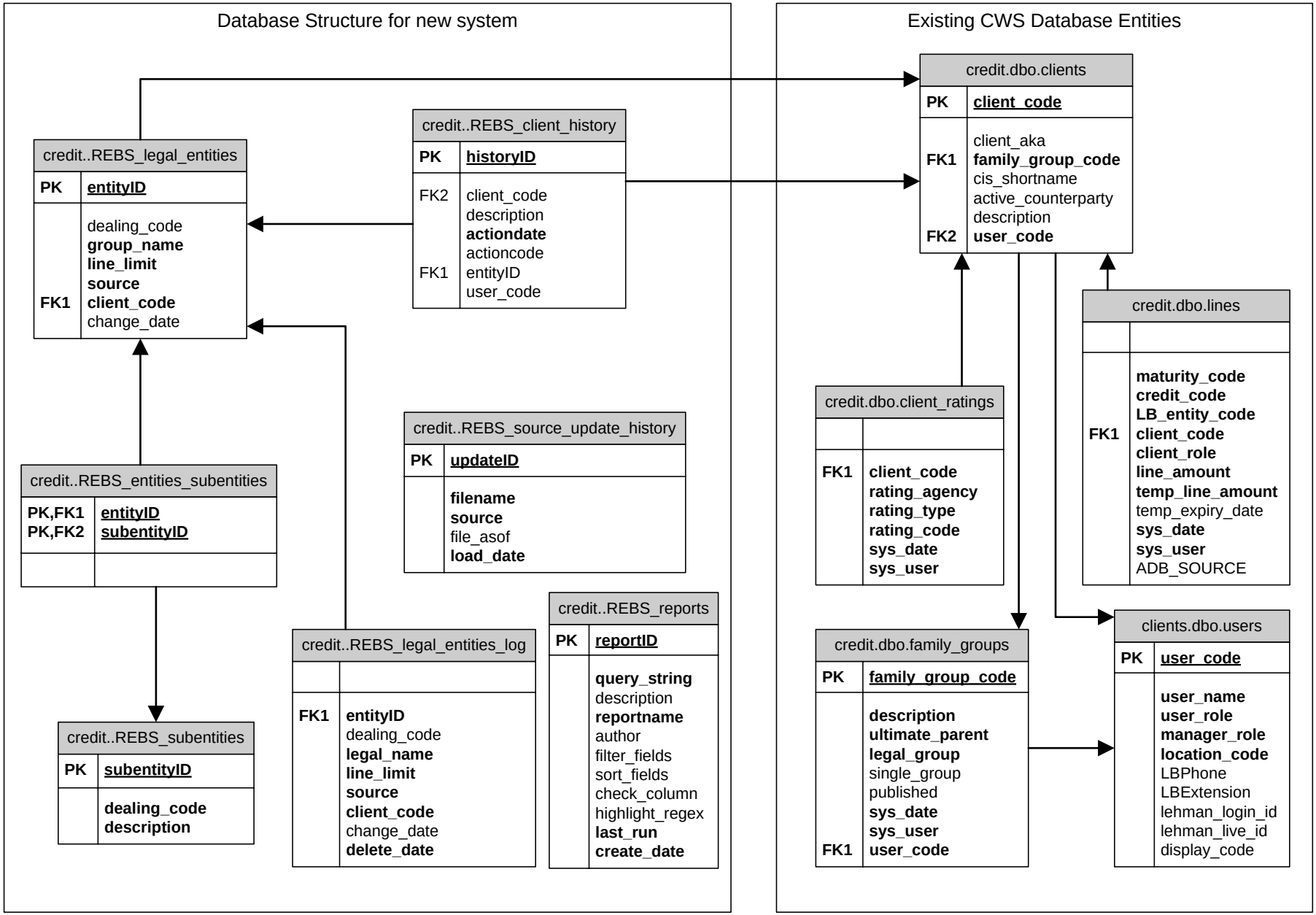
Not all information in this appendix can be shown due to confidentiality reasons. All sensitive information has been kept within the Lehman Brothers company.

	IR Swap		Swaption		xCCY swap (one leg only)
Cntry_Rsk_ID	14752	14753	31421	31422	16572
Product_Classification	fxs	fxs	fxs	fxs	fxs
Source_Deal_Ref	100404L	100404L	1034301L	1034301L	180184E
Source_Leg_Ref	100404L.63780	100404L.63781	1034301L.763N6T	1034301L.763N6V	180184E.186444
Portfolio	ZAR500	ZAR500	TWDFXIRSWP	TWDFXIRSWP	PLN592
Trade_Prod_ID	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
Prim_Prod_ID	1705430	1705430	1705430	1705430	1705430
Source_Prim_Prod_ID	USD	USD	USD	USD	USD
Prim_Global_Account_ID	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
Prim_Account_Name	ZAR500	ZAR500	TWDFXIRSWPN	TWDFXIRSWPN	PLN592
Cpty_Account_Number	79280CSFL	79280CSFL	032498CBAS	032498CBAS	121396LASI
Prim_Account_Number	72155	72155	89337	89337	18213
Deal_ID	0	0	0	0	0
Product_Group	FX	FX	FX	FX	FX
Source_System	SUM	SUM	SUM	SUM	SUM
Start_Date	19980421	19980421	20050928	20050928	20010704
End_Date	20080421	20080421	20101001	20101001	20060706
Report_Date	20051202	20051202	20051202	20051202	20051202
Security_Country_Code	ZA	ZA	TW	TW	PL
Asset_Class	D	D	D	D	D
Currency_Type	LH	LH	LH	LH	LH
Maturity_Bucket	6	6	6	6	4
Maturity_Code	2Y-5Y	2Y-5Y	2Y-5Y	2Y-5Y	6M-12M
Long_Short	S	L	L	L	S
Put_Call	P	P	P	P	P
Funding_Ccy	USD	USD	USD	USD	USD
Option_Ccy_Type	USD	USD	USD	USD	USD

Table E – Credit Risk Product comparison table

Appendix 2-B: Diagrams

These are the diagrams. The following diagram is an entity relationship diagram of the database structure.


Database Structure for new system
Existing CWS Database Entities
credit..REBS_legal_entities
PK entityID
dealing_code
group_name
line_limit
source
FK1 client_code
change_date
credit..REBS_client_history
PK historyID
FK2 client_code
description
actiondate
actioncode
FK1 entityID
user_code
credit..REBS_entities_subentities
PK,FK1 entityID
PK,FK2 subentityID
credit..REBS_source_update_history
PK updateID
filename
source
file_asof
load_date
credit..REBS_subentities
PK subentityID
dealing_code
description
credit..REBS_legal_entities_log
FK1 entityID
dealing_code
legal_name
line_limit
source
client_code
change_date
delete_date
credit..REBS_reports
PK reportID
query_string
description
reportname
author
filter_fields
sort_fields
check_column
highlight_regex
last_run
create_date
credit.dbo.clients
PK client_code
client_aka
FK1 family_group_code
cis_shortname
active_counterparty
description
FK2 user_code
credit.dbo.lines
FK1 maturity_code
credit_code
LB_entity_code
client_code
client_role
line_amount
temp_line_amount
temp_expiry_date
sys_date
sys_user
ADB_SOURCE
credit.dbo.client_ratings
FK1 client_code
rating_agency
rating_type
rating_code
sys_date
sys_user
credit.dbo.family_groups
PK family_group_code
description
ultimate_parent
legal_group
single_group
published
sys_date
sys_user
FK1 user_code
clients.dbo.users
PK user_code
user_name
user_role
manager_role
location_code
LBPhone
LBExtension
lehman_login_id
lehman_live_id
display_code

Appendix 2-C: Database Implementation Details

This section will list the SQL DDL code written during this project. These may be proprietary, but we will censor any sensitive information as it will be needed for Lehman Brothers employees.

Tables:

```
CREATE TABLE dbo.REBS_subentities
(
    subentityID  ROW_ID       IDENTITY,
    dealing_code varchar(4)   NOT NULL,
    description  varchar(128) NOT NULL
)

CREATE TABLE dbo.REBS_source_update_history
(
    updateID  ROW_ID       IDENTITY,
    filename  varchar(255) NOT NULL,
    source    varchar(32)  NOT NULL,
    file_asof datetime     NULL,
    load_date DATESTAMP    DEFAULT GETDATE() NOT NULL
)

CREATE TABLE dbo.REBS_reports
(
    reportID        numeric(18,0) IDENTITY,
    query_string    varchar(512)  NOT NULL,
    description     varchar(512)  NULL,
    reportname      varchar(32)   NOT NULL,
    author          varchar(32)   NULL,
    filter_fields   varchar(255)  NULL,
    sort_fields     varchar(255)  NULL,
    highlight_regex varchar(64)   DEFAULT '' NOT NULL,
    check_column    varchar(64)   DEFAULT '' NOT NULL,
    last_run        datetime      DEFAULT GETDATE() NOT NULL,
    create_date     datetime      DEFAULT GETDATE() NOT NULL,
    CONSTRAINT REBS_repor_21316427872
    PRIMARY KEY NONCLUSTERED (reportID)
)

CREATE TABLE dbo.REBS_legal_entities_log
(
    entityID     numeric(38,0) NOT NULL,
    dealing_code varchar(5)    NULL,
    group_name   varchar(256)  NOT NULL,
    line_amount  float         DEFAULT 0.0 NOT NULL,
    source       varchar(12)   NOT NULL,
    client_code  ID            NOT NULL,
    change_date  DATESTAMP     DEFAULT GETDATE() NOT NULL,
    action_code  varchar(16)   NULL
)

CREATE TABLE dbo.REBS_legal_entities
(
    entityID     ROW_ID       IDENTITY,
    dealing_code varchar(5)   NULL,
    group_name   varchar(256) NOT NULL,
    line_amount  float        DEFAULT 0.0 NOT NULL,
    source       varchar(12)  NOT NULL,
    client_code  ID           NOT NULL,
    change_date  DATESTAMP    NOT NULL
)
EXEC sp_bindefault 'dbo.current_date','REBS_legal_entities.change_date'
CREATE TRIGGER dbo.REBS_deleted_entity_history
ON dbo.REBS_legal_entities
FOR DELETE AS
INSERT INTO dbo.REBS_legal_entities_log (
               entityID,
               dealing_code,
               group_name,
               line_amount,
               source,
               client_code,
               change_date,
               action_code)
        SELECT entityID, dealing_code, group_name, line_amount, source, client_code,
change_date, 'DELETE'
        FROM deleted
```

```
DECLARE @entityID int
DECLARE @action_code varchar (32)
DECLARE @description varchar(512)
DECLARE @client_code ID

SELECT @entityID=entityID, @client_code = client_code FROM deleted
SET @action_code = 'DELETED'
SET @description = 'DELETED entity'

EXEC REBS_record_client_history @entityID,
               @action_code,
               @description,
               @client_code

CREATE TRIGGER dbo.REBS_inserted_entity_history
ON dbo.REBS_legal_entities
FOR INSERT AS
INSERT INTO dbo.REBS_legal_entities_log (
               entityID,
               dealing_code,
               group_name,
               line_amount,
               source,
               client_code,
               change_date,
               action_code)
       SELECT entityID, dealing_code, group_name, line_amount, source, client_code,
change_date, 'INSERT'
       FROM deleted

DECLARE @entityID int
DECLARE @action_code varchar (32)
DECLARE @description varchar(512)
DECLARE @client_code ID
SELECT @entityID=entityID, @client_code = client_code FROM inserted
SET @action_code = 'INSERTED'
SET @description = 'INSERTED entity'

EXEC REBS_record_client_history @entityID,
               @action_code,
               @description,
               @client_code

CREATE TRIGGER dbo.REBS_updated_entity_history
ON dbo.REBS_legal_entities
FOR UPDATE AS
       -- RECORD CHANGE INTO LOG
       INSERT INTO dbo.REBS_legal_entities_log (
               entityID,
               dealing_code,
               group_name,
               line_amount,
               source,
               client_code,
               change_date,
               action_code)
               SELECT entityID, dealing_code, group_name, line_amount, source,
client_code, change_date, 'UPDATE'
                       FROM inserted

       DECLARE @entityID int
       DECLARE @action_code varchar (32)
       DECLARE @description varchar(512)
       DECLARE @client_code ID

       DECLARE @limit_before float
       DECLARE @limit_after float

       DECLARE @old_client_code ID

       SELECT @old_client_code = client_code, @limit_before = line_amount FROM deleted
       SELECT @entityID=entityID, @client_code = client_code, @limit_after = line_amount
FROM inserted

       UPDATE dbo.REBS_legal_entities SET change_date = GETDATE() WHERE entityID =
@entityID

       IF (@old_client_code <> @client_code)
       BEGIN
               SET @action_code = 'UPDATED'
               SET @description = 'UPDATED client_code FROM ' + convert(varchar(10),
@old_client_code) + ' TO ' +
                       convert(varchar(10), @client_code)

               EXEC REBS_record_client_history @entityID,
                               @action_code,
                               @description,
```

```
                                @client_code
        END
        ELSE IF (@limit_before <> @limit_after)
        BEGIN
                SET @action_code = 'UPDATED'
                SET @description = 'UPDATED line_amount FROM ' + convert(varchar(10),
@limit_before) + ' TO ' +
                        convert(varchar(10), @limit_after)

                EXEC REBS_record_client_history @entityID,
                                @action_code,
                                @description,
                                @client_code
        END

CREATE TABLE dbo.REBS_entities_subentities
(
    entityID    numeric(38,0) NOT NULL,
    subentityID numeric(38,0) NOT NULL
)

CREATE TABLE dbo.REBS_client_history
(
    historyID   numeric(18,0) IDENTITY,
    client_code ID            NULL,
    entityID    ID            NULL,
    description varchar(512)  NULL,
    actiondate  DATESTAMP     NOT NULL,
    action_code varchar(32)   NOT NULL,
    user_code   MNEM10        NULL
)
```

Views:

```
CREATE VIEW dbo.REBS_client_history_view
AS
    SELECT credit.dbo.REBS_client_history.historyID,
credit.dbo.REBS_client_history.client_code, credit.dbo.REBS_client_history.entityID,
credit.dbo.REBS_client_history.description, credit.dbo.REBS_client_history.actiondate,
credit.dbo.REBS_client_history.action_code
,
    credit.dbo.clients.description AS cws_name,
    credit.dbo.REBS_legal_entities.group_name,
    credit.dbo.REBS_client_history.user_code
        FROM credit.dbo.REBS_client_history
                LEFT JOIN credit.dbo.clients ON credit.dbo.REBS_client_history.client_code
= credit.dbo.clients.client_code
                LEFT JOIN credit.dbo.REBS_legal_entities ON
credit.dbo.REBS_client_history.entityID = credit.dbo.REBS_legal_entities.entityID

CREATE VIEW REBS_dealing_codes
AS
SELECT REBS_legal_entities.dealing_code, REBS_legal_entities.entityID, null as
subentityID, REBS_legal_entities.group_name as description
FROM REBS_legal_entities WHERE REBS_legal_entities.dealing_code is not null
UNION
SELECT REBS_subentities.dealing_code, MIN(REBS_entities_subentities.entityID),
REBS_subentities.subentityID, REBS_subentities.description
    FROM REBS_subentities LEFT JOIN REBS_entities_subentities ON
REBS_subentities.subentityID = REBS_entities_subentities.subentityID
    GROUP BY REBS_subentities.dealing_code, REBS_subentities.subentityID,
REBS_subentities.description

CREATE VIEW REBS_existinglimits
AS
SELECT
    credit.dbo.clients.description,
    credit.dbo.REBS_legal_entities.client_code,
    EBSLON = convert( float, sum(REBS_legal_entities.line_amount * sign( patindex(
"EBSLON", rtrim(source))) )),
    EBSNYC = convert( float, sum(REBS_legal_entities.line_amount * sign( patindex(
"EBSNYC", rtrim(source))) )),
    EBSTOK = convert( float, sum(REBS_legal_entities.line_amount * sign( patindex(
"EBSTOK", rtrim(source))) )),
    REUTERS = convert( float, sum(REBS_legal_entities.line_amount * sign( patindex(
"REUTERS", rtrim(source))) )),
    totalLimit = sum(REBS_legal_entities.line_amount),
    CWSlimit = CASE WHEN (max(lines.temp_line_amount) > 0) THEN convert(float,
max(temp_line_amount)) ELSE convert(float, max(lines.line_amount)/1000.0) END,
    status = CASE WHEN (sum(REBS_legal_entities.line_amount) > ( CASE WHEN
(max(lines.temp_line_amount) > 0) THEN convert(float, max(temp_line_amount)) ELSE
convert(float, max(lines.line_amount)/1000.0) END)) THEN 'OVERLIMIT' ELSE '' END,
    recordCount = count(*)
FROM
    credit.dbo.REBS_legal_entities, credit.dbo.lines,
    credit.dbo.clients
WHERE
    credit.dbo.REBS_legal_entities.client_code = credit.dbo.lines.client_code AND
```

```
    credit.dbo.clients.client_code = credit.dbo.lines.client_code AND
    maturity_code = 'SPOT' AND
    credit_code = 'FX' AND
    lines.LB_entity_code = 'LBI'
GROUP BY
    credit.dbo.REBS_legal_entities.client_code,
    credit.dbo.clients.description,
    credit.dbo.lines.maturity_code,
    credit.dbo.lines.credit_code,
    credit.dbo.lines.LB_entity_code

CREATE VIEW REBS_fxspotclients
AS
SELECT lines.client_code, clients.description, lines.line_amount
FROM
    lines, clients
WHERE
    lines.client_code = clients.client_code AND
    lines.maturity_code = 'SPOT' AND
    lines.credit_code = 'FX' AND
    lines.LB_entity_code = 'LBI'

CREATE VIEW dbo.REBS_nonlinkedlimits
AS
    SELECT credit.dbo.REBS_legal_entities.entityID,
               credit.dbo.REBS_legal_entities.group_name,
               credit.dbo.REBS_legal_entities.line_amount,
               credit.dbo.REBS_legal_entities.source,
               credit.dbo.REBS_legal_entities.client_code
      FROM credit.dbo.REBS_legal_entities
     WHERE credit.dbo.REBS_legal_entities.client_code not in (select client_code from
clients)

CREATE VIEW REBS_otherlimits
AS
SELECT
    credit.dbo.REBS_legal_entities.client_code,
    limitEBSLON = convert( float, sum(line_amount * sign( patindex( "EBSLON",
rtrim(source))) )),
    limitEBSNYC = convert( float, sum(line_amount * sign( patindex( "EBSNYC",
rtrim(source))) )),
    limitEBSTOK = convert( float, sum(line_amount * sign( patindex( "EBSTOK",
rtrim(source))) )),
    limitREUTERS = convert( float, sum(line_amount * sign( patindex( "REUTERS",
rtrim(source))) )),
    totalLimit = sum(line_amount),
    recordCount = count(*)
FROM
    credit.dbo.REBS_legal_entities, credit.dbo.clients
WHERE
    credit.dbo.REBS_legal_entities.client_code = clients.client_code
GROUP BY
    credit.dbo.REBS_legal_entities.client_code

CREATE VIEW dbo.REBS_valid_groups
AS
               SELECT REBS_legal_entities.entityID, REBS_legal_entities.dealing_code,
REBS_legal_entities.group_name, REBS_legal_entities.line_amount,
REBS_legal_entities.source, REBS_legal_entities.client_code,
REBS_legal_entities.change_date
, credit.dbo.REBS_fxspotclients.description, credit.dbo.REBS_fxspotclients.line_amount as
cws_line_amount
               FROM credit.dbo.REBS_legal_entities LEFT JOIN credit.dbo.REBS_fxspotclients
                              ON credit.dbo.REBS_legal_entities.client_code =
credit.dbo.REBS_fxspotclients.client_code
               WHERE credit.dbo.REBS_fxspotclients.line_amount IS NOT null
```

Triggers:

```
IF OBJECT_ID('dbo.REBS_deleted_entity_history') IS NOT NULL
BEGIN
    DROP TRIGGER dbo.REBS_deleted_entity_history
    IF OBJECT_ID('dbo.REBS_deleted_entity_history') IS NOT NULL
        PRINT '<<< FAILED DROPPING TRIGGER dbo.REBS_deleted_entity_history >>>'
    ELSE
        PRINT '<<< DROPPED TRIGGER dbo.REBS_deleted_entity_history >>>'
END
go
CREATE TRIGGER dbo.REBS_deleted_entity_history
ON dbo.REBS_legal_entities
FOR DELETE AS
INSERT INTO dbo.REBS_legal_entities_log (
               entityID,
               dealing_code,
               group_name,
               line_amount,
               source,
               client_code,
```

```
                change_date,
                action_code)
        SELECT entityID, dealing_code, group_name, line_amount, source, client_code,
change_date, 'DELETE'
        FROM deleted


DECLARE @entityID int
DECLARE @action_code varchar (32)
DECLARE @description varchar(512)
DECLARE @client_code ID

SELECT @entityID=entityID, @client_code = client_code FROM deleted
SET @action_code = 'DELETED'
SET @description = 'DELETED entity'

EXEC REBS_record_client_history @entityID,
                @action_code,
                @description,
                @client_code
go
IF OBJECT_ID('dbo.REBS_deleted_entity_history') IS NOT NULL
    PRINT '<<< CREATED TRIGGER dbo.REBS_deleted_entity_history >>>'
ELSE
    PRINT '<<< FAILED CREATING TRIGGER dbo.REBS_deleted_entity_history >>>'
go

IF OBJECT_ID('dbo.REBS_updated_entity_history') IS NOT NULL
BEGIN
    DROP TRIGGER dbo.REBS_updated_entity_history
    IF OBJECT_ID('dbo.REBS_updated_entity_history') IS NOT NULL
        PRINT '<<< FAILED DROPPING TRIGGER dbo.REBS_updated_entity_history >>>'
    ELSE
        PRINT '<<< DROPPED TRIGGER dbo.REBS_updated_entity_history >>>'
END
go
CREATE TRIGGER dbo.REBS_updated_entity_history
ON dbo.REBS_legal_entities
FOR UPDATE AS
        -- RECORD CHANGE INTO LOG
        INSERT INTO dbo.REBS_legal_entities_log (
                entityID,
                dealing_code,
                group_name,
                line_amount,
                source,
                client_code,
                change_date,
                action_code)
                SELECT entityID, dealing_code, group_name, line_amount, source,
client_code, change_date, 'UPDATE'
                        FROM inserted

        DECLARE @entityID int
        DECLARE @action_code varchar (32)
        DECLARE @description varchar(512)
        DECLARE @client_code ID

        DECLARE @limit_before float
        DECLARE @limit_after float

        DECLARE @old_client_code ID

        SELECT @old_client_code = client_code, @limit_before = line_amount FROM deleted
        SELECT @entityID=entityID, @client_code = client_code, @limit_after = line_amount
FROM inserted

        UPDATE dbo.REBS_legal_entities SET change_date = GETDATE() WHERE entityID =
@entityID

        IF (@old_client_code <> @client_code)
        BEGIN
                SET @action_code = 'UPDATED'
                SET @description = 'UPDATED client_code FROM ' + convert(varchar(10),
@old_client_code) + ' TO ' +
                        convert(varchar(10), @client_code)

                EXEC REBS_record_client_history @entityID,
                                @action_code,
                                @description,
                                @client_code
        END
        ELSE IF (@limit_before <> @limit_after)
        BEGIN
                SET @action_code = 'UPDATED'
                SET @description = 'UPDATED line_amount FROM ' + convert(varchar(10),
@limit_before) + ' TO ' +
                        convert(varchar(10), @limit_after)
```

```
                EXEC REBS_record_client_history @entityID,
                                @action_code,
                                @description,
                                @client_code
        END
go
IF OBJECT_ID('dbo.REBS_updated_entity_history') IS NOT NULL
    PRINT '<<< CREATED TRIGGER dbo.REBS_updated_entity_history >>>'
ELSE
    PRINT '<<< FAILED CREATING TRIGGER dbo.REBS_updated_entity_history >>>'
go

IF OBJECT_ID('dbo.REBS_inserted_entity_history') IS NOT NULL
BEGIN
    DROP TRIGGER dbo.REBS_inserted_entity_history
    IF OBJECT_ID('dbo.REBS_inserted_entity_history') IS NOT NULL
        PRINT '<<< FAILED DROPPING TRIGGER dbo.REBS_inserted_entity_history >>>'
    ELSE
        PRINT '<<< DROPPED TRIGGER dbo.REBS_inserted_entity_history >>>'
END
go
CREATE TRIGGER dbo.REBS_inserted_entity_history
ON dbo.REBS_legal_entities
FOR INSERT AS
INSERT INTO dbo.REBS_legal_entities_log (
                entityID,
                dealing_code,
                group_name,
                line_amount,
                source,
                client_code,
                change_date,
                action_code)
        SELECT entityID, dealing_code, group_name, line_amount, source, client_code,
change_date, 'INSERT'
        FROM deleted

DECLARE @entityID int
DECLARE @action_code varchar (32)
DECLARE @description varchar(512)
DECLARE @client_code ID
SELECT @entityID=entityID, @client_code = client_code FROM inserted
SET @action_code = 'INSERTED'
SET @description = 'INSERTED entity'

EXEC REBS_record_client_history @entityID,
                @action_code,
                @description,
                @client_code
go
IF OBJECT_ID('dbo.REBS_inserted_entity_history') IS NOT NULL
    PRINT '<<< CREATED TRIGGER dbo.REBS_inserted_entity_history >>>'
ELSE
    PRINT '<<< FAILED CREATING TRIGGER dbo.REBS_inserted_entity_history >>>'
```

Stored procedures:

```
IF OBJECT_ID('dbo.REBS_run_report_by_id') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_run_report_by_id
    IF OBJECT_ID('dbo.REBS_run_report_by_id') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_run_report_by_id >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_run_report_by_id >>>'
END
go
CREATE PROCEDURE dbo.REBS_run_report_by_id
                @reportID varchar(10),
                @whereString varchar(1024) = '',
                @orderBy varchar(255) = null
AS
    BEGIN
        DECLARE @query_string varchar(512)
        DECLARE @orderByString varchar(255)
        DECLARE @defaultOrderField varchar(255)

        SELECT @query_string = query_string, @defaultOrderField = sort_fields
        FROM credit.dbo.REBS_reports
        WHERE reportID = convert(int, @reportID)

        IF (@orderBy is not null AND @orderBy != '')
        BEGIN
                        SET @orderByString = 'ORDER BY ' + @orderBy
        END
        ELSE
        BEGIN
                        -- SET DEFUALT SORT FIELD
                        IF (LEN(@defaultOrderField) > 0)
```

```
                        BEGIN
                                        SET @orderByString = 'ORDER BY ' +
@defaultOrderField
                        END
        END

        IF (@query_string is not null)
        BEGIN
                UPDATE REBS_reports SET last_run = GETDATE() WHERE reportID = convert(int,
@reportID)
                EXECUTE (
                @query_string + ' ' + @whereString + ' ' + @orderByString
                )
        END
    END




go
EXEC sp_procxmode 'dbo.REBS_run_report_by_id','unchained'
go
IF OBJECT_ID('dbo.REBS_run_report_by_id') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_run_report_by_id >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_run_report_by_id >>>'
go

IF OBJECT_ID('dbo.REBS_record_subentity') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_record_subentity
    IF OBJECT_ID('dbo.REBS_record_subentity') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_record_subentity >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_record_subentity >>>'
END
go
CREATE PROCEDURE dbo.REBS_record_subentity
                @entID varchar(10),
                @dealing_code varchar(4),
                @description varchar(128)
AS
    BEGIN
                DECLARE @entityID int
                DECLARE @entityIDexists int
                DECLARE @relation_exists int
                SET @entityID = convert(int, @entID)

                DECLARE @subentityID int

                SELECT @entityIDexists=COUNT(*)
                                FROM credit.dbo.REBS_legal_entities
                                WHERE entityID = @entityID

                IF (@entityIDexists > 0)
                BEGIN

                        SELECT @subentityID = subentityID FROM credit.dbo.REBS_subentities
                                WHERE   credit.dbo.REBS_subentities.dealing_code =
@dealing_code

                        -- if we cannot find the subentity, make a new row for it
                        IF (@subentityID is null OR @subentityID < 0)
                        BEGIN
                                        INSERT INTO credit.dbo.REBS_subentities
(dealing_code, description)
                                                VALUES (@dealing_code, @description)

                                        SELECT @subentityID = subentityID FROM
credit.dbo.REBS_subentities
                                                WHERE
        credit.dbo.REBS_subentities.dealing_code = @dealing_code
                        END

                        SELECT @relation_exists=COUNT(*) FROM
credit.dbo.REBS_entities_subentities WHERE entityID = @entityID AND subentityID =
@subentityID

                        IF (@relation_exists = 0)
                        BEGIN
                                        INSERT INTO credit.dbo.REBS_entities_subentities
(entityID, subentityID)
                                                        VALUES (@entityID, @subentityID)
                        END
                END

                SELECT @subentityID AS subentityID
    END
```

```
go
EXEC sp_procxmode 'dbo.REBS_record_subentity','unchained'
go
IF OBJECT_ID('dbo.REBS_record_subentity') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_record_subentity >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_record_subentity >>>'
go

IF OBJECT_ID('dbo.REBS_record_source_update') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_record_source_update
    IF OBJECT_ID('dbo.REBS_record_source_update') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_record_source_update >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_record_source_update >>>'
END
go
CREATE PROCEDURE dbo.REBS_record_source_update
               @filename varchar(255),
               @source varchar(32),
               @file_asof datetime = null
AS
    BEGIN
        INSERT INTO credit.dbo.REBS_source_update_history (filename, source, file_asof)
        VALUES (@filename, @source, @file_asof)
    END



go
EXEC sp_procxmode 'dbo.REBS_record_source_update','unchained'
go
IF OBJECT_ID('dbo.REBS_record_source_update') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_record_source_update >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_record_source_update >>>'
go

IF OBJECT_ID('dbo.REBS_record_entity') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_record_entity
    IF OBJECT_ID('dbo.REBS_record_entity') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_record_entity >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_record_entity >>>'
END
go
CREATE PROCEDURE dbo.REBS_record_entity
               @source varchar(12),
               @entity_name varchar(256),
               @line_limit varchar(10),
               @dealing_code varchar(5) = '',
               @client_code ID = -1
AS
    BEGIN
       DECLARE @rtnstring varchar(1024)
       DECLARE @action_taken varchar(20)
       DECLARE @existID int
       DECLARE @limit float
       DECLARE @known_client_code ID
       SET @known_client_code = -1
       -- convert the limit from a string to an actual float limit
       SET @limit = CONVERT(float, @line_limit)

       -- query tablet to see if entity and source already exist
       SELECT @existID=entityID FROM credit.dbo.REBS_legal_entities
              WHERE group_name = @entity_name AND source = @source

       SET @rtnstring = CONVERT(varchar(5), @existID) + '; '

       -- if the source ID doesn't exist
       IF (@existID is null)
       BEGIN
              SET @rtnstring = @rtnstring + 'no ID exists; '

              IF (@client_code = -1)
              BEGIN
                     SET @rtnstring = @rtnstring + 'no client code entered; '

                     SELECT @known_client_code=client_code FROM
credit.dbo.REBS_legal_entities
                            WHERE group_name = @entity_name AND client_code > 0

                     -- check to see if a name->client_code matching already exists
```

```
                        IF (@known_client_code is not null)
                        BEGIN
                                SET @rtnstring = @rtnstring + 'client code already
exists for another source; '
                                -- insert the counterparty name into the database
not linked to a CWS client_code
                                SET @client_code = @known_client_code
                        END
                END

                INSERT INTO dbo.REBS_legal_entities ( dealing_code, group_name,
line_amount, source, client_code )
                                VALUES (
                                        @dealing_code,
                                        @entity_name,
                                        @limit,
                                        @source,
                                        @client_code)
                        SET @action_taken = 'INSERT'
        END
        ELSE            --AN entity ID already exists in the table
        BEGIN
                        IF (@client_code = -1)
                        BEGIN
                                SELECT @known_client_code=client_code FROM
credit.dbo.REBS_legal_entities
                                        WHERE group_name = @entity_name AND source = @source

                                -- check to see if a name->client_code matching already
exists
                                IF (@known_client_code is not null AND @known_client_code >
0)
                                BEGIN
                                        SET @client_code = @known_client_code
                                END
                                ELSE
                                BEGIN
                                        SELECT @known_client_code=client_code FROM
credit.dbo.REBS_legal_entities
                                                WHERE group_name = @entity_name AND
client_code > 0
                                END

                                IF (@known_client_code > 0)
                                BEGIN
                                        SET @client_code = @known_client_code
                                END
                        END

                        -- UPDATE the existing entry with new limit
                        UPDATE dbo.REBS_legal_entities SET
                                        dealing_code = @dealing_code,
                                        line_amount = @limit,
                                        change_date = GETDATE(),
                                        client_code = @client_code
                                WHERE
                                        group_name = @entity_name AND source = @source
                        SET @action_taken = 'UPDATE'
        END

        SELECT @existID=entityID FROM credit.dbo.REBS_legal_entities
                WHERE group_name = @entity_name AND source = @source

        SELECT  @existID,
                        @action_taken AS action_taken,
                        @source,
                                @entity_name,
                                @limit,
                                @dealing_code,
                                @client_code ID,
                                isnull(@known_client_code, 123456),
                                @rtnstring as Message
        --SELECT isnull(@existID, 0) AS entityID, 12 as client_code

        RETURN isnull(@existID, 0)
    END




go
EXEC sp_procxmode 'dbo.REBS_record_entity','unchained'
go
IF OBJECT_ID('dbo.REBS_record_entity') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_record_entity >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_record_entity >>>'
go
```

```
IF OBJECT_ID('dbo.REBS_record_client_history') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_record_client_history
    IF OBJECT_ID('dbo.REBS_record_client_history') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_record_client_history >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_record_client_history >>>'
END
go
CREATE PROCEDURE dbo.REBS_record_client_history
               @entityID int = -1,
               @action_code varchar (32),
               @description varchar(512),
               @client_code ID = -1,
               @user_code varchar(10) = 'sys'
AS
    BEGIN
               INSERT INTO REBS_client_history (client_code, entityID, description,
actiondate, action_code, user_code)
                              VALUES (@client_code, @entityID, @description, getdate(),
@action_code, @user_code)
    END


go
EXEC sp_procxmode 'dbo.REBS_record_client_history','unchained'
go
IF OBJECT_ID('dbo.REBS_record_client_history') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_record_client_history >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_record_client_history >>>'
go

IF OBJECT_ID('dbo.REBS_entity_subentities') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_entity_subentities
    IF OBJECT_ID('dbo.REBS_entity_subentities') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_entity_subentities >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_entity_subentities >>>'
END
go
CREATE PROCEDURE dbo.REBS_entity_subentities
               @entityID varchar(10) = null
AS
    BEGIN
       IF (@entityID is not null AND @entityID <> '')
       BEGIN
               SELECT REBS_subentities.dealing_code,
                      REBS_subentities.description,
                      REUTERS = convert( int, sum(1 * sign( patindex( "REUTERS",
rtrim(source))) )),
                      EBSLON = convert( int, sum(1 * sign( patindex( "EBSLON",
rtrim(source))) )),
                      EBSNYC = convert( int, sum(1 * sign( patindex( "EBSNYC",
rtrim(source))) )),
                      EBSTOK = convert( int, sum(1 * sign( patindex( "EBSTOK",
rtrim(source))) ))
                      FROM REBS_subentities
                             JOIN REBS_entities_subentities ON
REBS_subentities.subentityID = REBS_entities_subentities.subentityID
                             LEFT JOIN REBS_legal_entities ON
REBS_entities_subentities.entityID = REBS_legal_entities.entityID
                      WHERE REBS_legal_entities.entityID = CONVERT(int, @entityID)
                      GROUP BY REBS_subentities.dealing_code,
                             REBS_subentities.description
               END
    END




go
EXEC sp_procxmode 'dbo.REBS_entity_subentities','unchained'
go
IF OBJECT_ID('dbo.REBS_entity_subentities') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_entity_subentities >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_entity_subentities >>>'
go

IF OBJECT_ID('dbo.REBS_entity_details') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_entity_details
    IF OBJECT_ID('dbo.REBS_entity_details') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_entity_details >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_entity_details >>>'
```

```
END
go
CREATE PROCEDURE dbo.REBS_entity_details
              @eID varchar(10)
AS
    BEGIN
    DECLARE @entityID int
    SET @entityID = CONVERT(int, @eID)

    IF (@entityID is null OR @entityID < 0)
    BEGIN
              return
    END

    SELECT credit.dbo.REBS_legal_entities.entityID,
                  credit.dbo.REBS_legal_entities.group_name,
              credit.dbo.REBS_legal_entities.line_amount,
              credit.dbo.REBS_legal_entities.source,
              credit.dbo.REBS_legal_entities.client_code,
              credit.dbo.clients.description,
                     ISNULL(CASE WHEN (credit.dbo.lines.temp_line_amount > 0)
                                  THEN
                                     convert(float,
credit.dbo.lines.temp_line_amount)/1000.0
                                  ELSE
                                     convert(float, credit.dbo.lines.line_amount)/1000.0
                                  END, 0)
                     AS client_line_amount
              FROM credit.dbo.REBS_legal_entities
              LEFT JOIN credit.dbo.clients ON credit.dbo.REBS_legal_entities.client_code
= credit.dbo.clients.client_code
              LEFT JOIN credit.dbo.lines on credit.dbo.lines.client_code =
credit.dbo.REBS_legal_entities.client_code
              WHERE (entityID = @entityID) AND
              ((credit.dbo.lines.maturity_code = 'SPOT' AND
              credit.dbo.lines.credit_code = 'FX' AND
              credit.dbo.lines.LB_entity_code = 'LBI') OR (credit.dbo.lines.client_code
is null))
    END




go
EXEC sp_procxmode 'dbo.REBS_entity_details','unchained'
go
IF OBJECT_ID('dbo.REBS_entity_details') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_entity_details >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_entity_details >>>'
go

IF OBJECT_ID('dbo.REBS_cws_client_details') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_cws_client_details
    IF OBJECT_ID('dbo.REBS_cws_client_details') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_cws_client_details >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_cws_client_details >>>'
END
go
CREATE PROCEDURE dbo.REBS_cws_client_details
              @cc varchar(10)
AS
    BEGIN
              DECLARE @client_code ID
              SET @client_code = CONVERT(int, @cc)

              IF (@client_code is null OR @client_code <= 0)
              BEGIN
                        return
              END

              SELECT credit.dbo.clients.client_code,
              credit.dbo.clients.description,
              credit.dbo.clients.client_aka,
              credit.dbo.clients.user_code,
              credit.dbo.clients.physical_country_code,
              credit.dbo.clients.legal_country_code,
              credit.dbo.clients.industry_code,
              CASE WHEN (credit.dbo.lines.temp_line_amount > 0)
                        THEN
                                  convert(float,
credit.dbo.lines.temp_line_amount)/1000.0
                     WHEN (credit.dbo.lines.line_amount is null) THEN 0.0
                        ELSE
                                  convert(float,
credit.dbo.lines.line_amount)/1000.0
```

```
                              END AS line_amount
                        FROM credit.dbo.clients LEFT JOIN
                              credit.dbo.lines ON credit.dbo.clients.client_code =
credit.dbo.lines.client_code
                        WHERE credit.dbo.clients.client_code = @client_code AND
                              ((credit.dbo.lines.maturity_code = 'SPOT' AND
                              credit.dbo.lines.credit_code = 'FX' AND
                              credit.dbo.lines.LB_entity_code = 'LBI') OR
                     (credit.dbo.lines.maturity_code is null))
    END


go
EXEC sp_procxmode 'dbo.REBS_cws_client_details','unchained'
go
IF OBJECT_ID('dbo.REBS_cws_client_details') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_cws_client_details >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_cws_client_details >>>'
go

IF OBJECT_ID('dbo.REBS_client_subentities') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_client_subentities
    IF OBJECT_ID('dbo.REBS_client_subentities') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_client_subentities >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_client_subentities >>>'
END
go
CREATE PROCEDURE dbo.REBS_client_subentities
               @client_code varchar(10) = null
AS
    BEGIN
        IF (@client_code is not null AND @client_code <> '')
        BEGIN
               SELECT REBS_subentities.dealing_code,
                      REBS_subentities.description,
                      REUTERS = convert( int, sum(1 * sign( patindex( "REUTERS",
rtrim(source))) )),
                      EBSLON = convert( int, sum(1 * sign( patindex( "EBSLON",
rtrim(source))) )),
                      EBSNYC = convert( int, sum(1 * sign( patindex( "EBSNYC",
rtrim(source))) )),
                      EBSTOK = convert( int, sum(1 * sign( patindex( "EBSTOK",
rtrim(source))) ))
                      FROM REBS_subentities
                            JOIN REBS_entities_subentities ON
REBS_subentities.subentityID = REBS_entities_subentities.subentityID
                            LEFT JOIN REBS_legal_entities ON
REBS_entities_subentities.entityID = REBS_legal_entities.entityID
                      WHERE REBS_legal_entities.client_code = CONVERT(int, @client_code)
                      GROUP BY REBS_subentities.dealing_code,
                            REBS_subentities.description
               END
    END




go
EXEC sp_procxmode 'dbo.REBS_client_subentities','unchained'
go
IF OBJECT_ID('dbo.REBS_client_subentities') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_client_subentities >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_client_subentities >>>'
go

IF OBJECT_ID('dbo.REBS_assoc_entities') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_assoc_entities
    IF OBJECT_ID('dbo.REBS_assoc_entities') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_assoc_entities >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_assoc_entities >>>'
END
go
CREATE PROCEDURE dbo.REBS_assoc_entities
               @eID varchar(10)
AS
    BEGIN
        DECLARE @entityID int
        DECLARE @client_code ID
        DECLARE @msg varchar(1024)

        SET @entityID = CONVERT(int, @eID)

        SELECT @client_code=client_code FROM credit.dbo.REBS_legal_entities
```

```
                        WHERE credit.dbo.REBS_legal_entities.entityID = @entityID

        SET @msg = CONVERT(varchar(10), @client_code)

        IF (@client_code is not null AND @client_code >= 0)
        BEGIN
                SELECT credit.dbo.REBS_legal_entities.group_name,
                                credit.dbo.REBS_legal_entities.source,
                                credit.dbo.REBS_legal_entities.entityID,
                                credit.dbo.REBS_legal_entities.line_amount
                        FROM credit.dbo.REBS_legal_entities
                        WHERE
                                        credit.dbo.REBS_legal_entities.client_code =
@client_code
        END
    END




go
EXEC sp_procxmode 'dbo.REBS_assoc_entities','unchained'
go
IF OBJECT_ID('dbo.REBS_assoc_entities') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_assoc_entities >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_assoc_entities >>>'
go

IF OBJECT_ID('dbo.REBS_link_entity') IS NOT NULL
BEGIN
    DROP PROCEDURE dbo.REBS_link_entity
    IF OBJECT_ID('dbo.REBS_link_entity') IS NOT NULL
        PRINT '<<< FAILED DROPPING PROCEDURE dbo.REBS_link_entity >>>'
    ELSE
        PRINT '<<< DROPPED PROCEDURE dbo.REBS_link_entity >>>'
END
go
CREATE PROCEDURE dbo.REBS_link_entity
                @entityID varchar(10),
                @client_code varchar(10),
                @user_code varchar(10)
AS
    BEGIN
        DECLARE @description varchar(255)
        DECLARE @eID int
        DECLARE @cc int

        SET @eID = convert(int, @entityID)
        SET @cc = convert(int, @client_code)
        SET @description = 'LINKED entity ID ' + @entityID + ' to client code ' +
@client_code
        EXEC REBS_record_client_history @eID,
                                'LINKED',
                                @description,
                                @cc,
                                @user_code

        UPDATE credit.dbo.REBS_legal_entities SET client_code = @cc WHERE entityID=@eID
        SELECT 'UPDATED' AS statuts
    END


go
EXEC sp_procxmode 'dbo.REBS_link_entity','unchained'
go
IF OBJECT_ID('dbo.REBS_link_entity') IS NOT NULL
    PRINT '<<< CREATED PROCEDURE dbo.REBS_link_entity >>>'
ELSE
    PRINT '<<< FAILED CREATING PROCEDURE dbo.REBS_link_entity >>>'
```

Appendix 2-D: Application Layer Implementation Details

The implementation details of the data importing Perl scripts are described below. The actual implementations can be found in use on the Lehman Brothers systems, under the CRM data importing scripts directory. The actual code is not shown here for (1) security reasons and (2) the length of the scripts is long.

Script	Purpose
`Dbaccess.pl`	Common functions and database connection details. All database connectivity is implemented in this class.
`readReuters.pl`	Perl script for reading the Reuters data feed file.
`readEBS.pl`	Perl script for reading the various EBS data files. The source location is derived from the filename: LEHL, LEHQ, LEHN for London, Tokyo and New York files, respectively.

The implementation details of the JSP interface that is part of the CRM web page are briefly described below. The actual implementations of these files can be found in the CRM source directory.

Script	Purpose
`Detail_client.jsp`	Detail view page for CWS client details.
`Detail_entity.jsp`	Detail view page for entity.
`Project_reports.jsp`	The index and main page for accessing details and other pages. This is the "portal" for everything in the interface.
`Rebs.css`	The cascading style sheet for the JSP interfaces implementation.
`Record_history.jsp`	Sub-frame for recording history entries for counterparties/entities.
`Update.jsp`	Sub-frame for linking/unlinking a given entity to/from a CWS counterparty.

Appendix 2-E: Meeting Minutes

The following are relevant meeting minutes.

WPI Credit Project Meeting

Monday, 24 October 2005: 17:00 – 18:00

Purpose: CWS, Reuters and EBS

Present: William Hays, Amy Jackson, Mahvish Ayoob

Notes:

Credit Workstation (CWS) can be found at: http://my.lehman.com/CRM

CWS is the Lehman Brothers internal credit analysis and reporting interface and database backend.

Contained in this is the total credit limits and credit information up to the previous date (Reuters and EBS).

- Client Tab -> Client File -> Lines: to look up counterparty limits by product. We are interested in FX LBI – the foreign exchange for Lehman Brothers. The credit limits are shown in the thousands.
- Exposure Tab -> Client -> Summary: shows the current exposure.

Credit Ratings

Credit ratings are either internal or external. External are either Moodys or Standard and Poors.

From these ratings and review appetite, analysts create credit limit numbers.

[Traveled to trading floor]

- Spot limits are either a no limit or a numerical limit
- Forward limits are either no limit or 'yes,' a limit.

- Limits are 'No' by default. The only way to tell if the 'No' decision for a limit was by an analyst or by default is by logging into Reuters or EBS on the trading floor.

WPI Credit Project Meeting

Monday, 24 October 2005: 11:30 – 12:30

Present: William Hays, Amy Jackson, Thomas McConnon, Mahvish Ayoob

Unavailable: Leesan Wong

Notes:

Spot and Forward Matching

Reuters

- Ms. Wong receives from Reuters three (3) different emails (on a weekly basis):
    - Attached Microsoft Excel spreadsheet containing office code, location, status, and number of potential trades for spot trades.
    - Attached Microsoft Excel spreadsheet containing office code, location, status, and number of potential trades for forward trades.
    - New client email (Microsoft Word document), containing a file listing new clients added for spot and forward trades.

Spot Matching

EBS

- Ms. Wong receives, by EBS (London), an email on a weekly basis with an attached Adobe Acrobat (PDF) file containing listing of offices (code, location, status) for spot trades:
    - Credit allocation list: customers currently on EBS (London) Spot system.
    - Customer list change: shows the change in customer details for coming week.
        - For both of the above files, an 'X' on the left-hand-side indicates a new client.
        - For both of the above files, the reason for a new account allocation or existing account change is described in the right-hand-side column.
    - Customer credit communications: information that customers have requested to be distributed.
    - Additional communications: any additional information.

Limit Allocation Numbers

- Limit allocation numbers come from the following:
    - Reuters – Global
    - EBS – New York
    - EBS – Tokyo
    - EBS – London
    - CWS – Lehman Brothers Global (LBI Limits)
- Compare Reuters and EBS office credit limit totals to the CWS limits (end result is to see if totals are within CWS limit).
- CWS limit are in place and change only when trader or analyst requests limit increase or decrease.
- *Reuter and EBS use the same client code (dealing code), CWS does not. Legal names might also differ.

Current Process

1. Ms. Wong receives email from Reuters (weekly) containing spot and forward matches for previous week.
2. Ms. Wong receives email from EBS (weekly) containing spot matches for previous week.
3. Ms. Wong forwards (1) and (2) to Ms. Ayoob.
4. Ms. Ayoob prioritizes the lists based on the number of requests for trades with the office.
5. [*OPTIONALLY*] Ms. Ayoob checks to see if a limit already exists for an office by checking Credit Workstation (CWS) or previous data table received containing Reuters/EBS current credit limits. The previous data table from Reuters/EBS is requested by Ms. Wong.
6. Ms. Ayoob then emails requests to the corresponding analysts for the offices, asking for the credit limit for a particular office. Depending on the urgency of the credit limit request, Ms.

Ayoob will process the following step if needed. The credit limit response from the analyst is based off credit rating (external and internal) and review appetite.

7. Ms. Ayoob then takes the credit limit (whether it is YES, NO, or a specific limit) and enters these into Reuters/EBS on the trading floor.
    a. The credit limit is added/deleted/increased/decreased or the group is added/removed/created/deleted following the documented procedure.

Expectations from our project

- Formulate an automated system for keeping track of credit limits for offices from EBS and Reuters, comparing these to the Lehman Brothers internal CWS (Credit Workstation) system to ensure that the sum of all EBS and Reuters limits are within the approved LBI Daily Settlement Line in CWS.
- Keep a log of clients that have been approved or disapproved by the analysts to minimize repeated work.
- (If time,) Make sure that groupings of branches with their head offices are correct. Also, make sure that credit limits that are set for head offices are the same for their full and/or sub branches.

Next Steps

- Go down to trading floor after hours to see last step of process (entering data into Reuters and EBS applications).
- Develop a process flow chart outlining the current process flows.
    - Dependency chart tying data dependencies to user processes.
- Analyze underlying database structure for CWS
    - How is the CWS forward set up? (meeting with Ms. Ayoob)
    - Discuss the underlying CWS database tables with Mr. Eisen.
- Get list of possible raw data formats from EBS/Reuters.
- Get list of possible raw data formats from EBS/Reuters.
- WPI Credit Project Meeting

Tuesday 25 October 2005: 14:00 – 15:00

Purpose: Discussion on Client Workstation (CWS) back tables

Present: Daniel Eisen, William Hays, Amy Jackson

Notes:

There are three main CWS databases:

1. Client database (most used)
    - Feeds client tab, which contains the counterparty information.
    - On the front end, limit structure; Product group (we're interested in emerging markets foreign exchange, EMGFX, and foreign exchange, FX), and Legal entity (we're interested in LBI/LBE). Broken down by time buckets. All limit numbers are stored in 1000s.
2. Exposure database
3. Aggregate Exposure database (less used)

Dorm= Dormant Rating

CWS is updated the previous day, t+1.

Principal vs. Agents: set of funds vs. funds manager. No legal structure, rather from a risk point of view. Principal are faced directly.
Thinking about matching up CWS tables with Reuters and EBS data via legal names, seeing different client codes are used.

Tables

CREDIT.DBO.CLIENTS	
FIELD NAME	*DESCRIPTION*
client_code	unique identifier for client (integer)
client_aka	Short abbreviation of the client name (usually a number of characters)
family_group_code	Family group foreign key used to identify clients' parent company/client.
user_code	The user code for the analyst (foreign key used to reference user details in credit.dbo.users table.
physical_country_code	Country that client is in.
legal_country_code	
state_code	State of client, if in USA.
industry_code	
description	Legal name for client
watch_status	
active_counterpart_ind	Y or N depending on whether or not the counterparty is active.
active_issuer_ind	
last_review_date	
next_review_date	
cis_shortname	Short name of client.
sys_date	
sys_user	
ecam_industry_group	
sic_code	
home_page_url	
margin_tier_id	
last_activity_date	
aum_amount	
aum_date	
aum_category_code	
reviewable_ind	
research_reason_code	

CREDIT.DBO.CLIENT_RATINGS	
FIELD NAME	*DESCRIPTION*
client_code	Identifier for client
rating_agency	Agency for which the rating is for. (ie. S&P, FITCH, MOODY, or CCNEW for internal rating)
rating_type	Type for the rating (ie. INTRN for internal rating)
rating_code	The actual rating
sys_date	Date of the rating
sys_user	

CREDIT.DBO.USERS	
FIELD NAME	*DESCRIPTION*
user_code	Unique identifier for user
user_name	Usually [Last name], [First name]
user_role	Short descriptor for user role
manager_code	Manager's role
location_code	Country indication (UK for Europe, NY for US, HK for Hong Kong, etc.)
LBPhone	Phone number
LBExtension	Extension
PagerNumber	Pager number
FaxNumber	Fax number
MobileNumber	Mobile phone number
HomePhone	Home phone number
sys_date	
sys_user	
lehman_login_id	
lehman_live_id	
display_code	

CREDIT.DBO.FAMILY_GROUPS	
FIELD NAME	*DESCRIPTION*
family_group_code	Unique integer identifier for family group.
description	Legal name for family group
ultimate_parent	Parent for group
legal_group	Y or N indicated whether legal group
single_group	
published	
owner_code	Users foreign key identifying group owner
sys_date	
sys_user	

CREDIT.DBO.LINES	
FIELD NAME	*DESCRIPTION*
maturity_code	Time indicator (ie. SPOT, M03 for month 3)
credit_code	The type of credit code (ie. FX for foreign exchange, EMGFX for emerging markets foreign exchange)
LB_entity_code	Lehman Brothers internal entity code (LBI for our purposes)
client_code	Client code foreign key to credit.dbo.client_code table.
client_role	PRINCIPAL or AGENT (fund or manager, respectively)
line_amount	The credit line limit (0 or number)
temp_line_amount	A temporary line limit that exists before the temp_expiry_date. Otherwise zero (0).
temp_expiry_date	Temporary line amount expiration date.
sys_date	
sys_user	
ADB_SOURCE	

Views

CREDIT.DBO.CWS_CLIENT_VIEW

```
SELECT
      cl.client_code,
      cl.description,
      cl.industry_code,
      ind.description,
      cl.legal_country_code,
      cnt.country_name,
      cl.physical_country_code,
      cl.watch_status,
      cr.rating_code,
      ISNULL(cl.user_code, '?UNK?'),
      ISNULL(usr.display_code, '?UNK?'),
      usr.location_code,
      usr.LBExtension,
      usr.manager_code,
      cl.last_review_date,
      cl.next_review_date,
      cl.aum_amount,
      cl.aum_date,
      cl.aum_category_code,
      aum.aum_category_name
FROM credit..clients cl (INDEX clients_pk),
      credit..client_ratings cr,
      credit..users usr,
      credit..ISO_countries cnt (INDEX ISO_countries_pk ),
      credit..industries ind,
      credit..cws_client_aum_categories aum
WHERE
      (cl.user_code *= usr.user_code )
      AND
      (cl.client_code *= cr.client_code
            AND  cr.rating_agency    = 'CCNEW'
            AND  cr.rating_type        = 'INTRN')
      AND
      (cl.legal_country_code = cnt.ISO_3)
      AND
      (cl.industry_code = ind.industry_code)
      AND
      (cl.aum_category_code *= aum.aum_category_code)
```

FIELD NAME	*DESCRIPTION*
client_code	String unique identifier for client.
client_name	Client legal name
industry_code	
industry_name	
legal_country_code	Short country code (ie USA)
legal_country_name	Long country name (ie. UNITED STATES)
physical_country_code	
watch_status	
internal_rating	Internal rating from Lines table
analyst_code	Analyst user code
analyst_display_code	
analyst_location_code	
analyst_phone_extension	
manager_code	Foreign key to users table

last_review_date	
next_review_date	
aum_amount	
aum_date	
aum_category_code	
aum_category_name	

CREDIT.DBO.LINES_CROSS_TABLE

```
select
       l.client_code,
       l.credit_code,
       l.LB_entity_code,
       l.client_role,

       CL        = convert( float, sum(line_amount      * sign(
patindex(  "C/L", rtrim(maturity_code))) )),
       CL_temp   = convert( float, sum(temp_line_amount * sign(
patindex(  "C/L", rtrim(maturity_code))) )),

       ONL       = convert( float, sum(line_amount * sign( patindex(
"O/N", rtrim(maturity_code))) )),
       ON_temp   = convert( float, sum(temp_line_amount * sign(
patindex(  "O/N", rtrim(maturity_code))) )),

       W01       = convert( float, sum(line_amount * sign( patindex(
"W01", rtrim(maturity_code))) )),
       W01_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "W01", rtrim(maturity_code))) )),

       W01_3     = convert( float, sum(line_amount      * sign(
patindex(  "W01-3", rtrim(maturity_code))) )),
       W01_3_temp = convert( float, sum(temp_line_amount * sign(
patindex(  "W01-3", rtrim(maturity_code))) )),

       DSL1      = convert( float, sum(line_amount * sign( patindex(
"DSL", rtrim(maturity_code))) )),
       DSL1_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "DSL", rtrim(maturity_code))) )),

       DSL2      = convert( float, sum(line_amount * sign( patindex(
"DSL2", rtrim(maturity_code))) )),
       DSL2_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "DSL2", rtrim(maturity_code))) )),

       DSL3      = convert( float, sum(line_amount * sign( patindex(
"DSL3", rtrim(maturity_code))) )),
       DSL3_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "DSL3", rtrim(maturity_code))) )),

       DSL4      = convert( float, sum(line_amount * sign( patindex(
"DSL4", rtrim(maturity_code))) )),
       DSL4_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "DSL4", rtrim(maturity_code))) )),

       DSL5      = convert( float, sum(line_amount * sign( patindex(
"DSL5", rtrim(maturity_code))) )),
       DSL5_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "DSL5", rtrim(maturity_code))) )),

       BORR      = convert( float, sum(line_amount * sign( patindex(
"BORR", rtrim(maturity_code))) )),
       BORR_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "BORR", rtrim(maturity_code))) )),

       LEND      = convert( float, sum(line_amount * sign( patindex(
"LEND", rtrim(maturity_code))) )),
       LEND_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "LEND", rtrim(maturity_code))) )),

       M01       = convert( float, sum(line_amount * sign( patindex(
"M01", rtrim(maturity_code))) )),
       M01_temp   = convert( float, sum(temp_line_amount * sign(
patindex(  "M01", rtrim(maturity_code))) )),

       SPOT      = convert( float, sum(line_amount * sign( patindex(
"SPOT", rtrim(maturity_code))) )),
       SPOT_temp = convert( float, sum(temp_line_amount * sign(
```

```
patindex(  "SPOT", rtrim(maturity_code))) ),

       M014       = convert( float, sum(line_amount * sign( patindex(
"M01-4", rtrim(maturity_code))) )),
       M014_temp  = convert( float, sum(temp_line_amount * sign(
patindex(  "M01-4", rtrim(maturity_code))) )),

       M1_25     = convert( float, sum(line_amount      * sign(
patindex(  "M1-25", rtrim(maturity_code))) )),
       M1_25_temp = convert( float, sum(temp_line_amount * sign(
patindex(  "M1-25", rtrim(maturity_code))) )),

       M03        = convert( float, sum(line_amount * sign( patindex(
"M03", rtrim(maturity_code))) )),
       M03_temp   = convert( float, sum(temp_line_amount * sign(
patindex(  "M03", rtrim(maturity_code))) )),

       M06      = convert( float, sum(line_amount * sign( patindex(
"M06", rtrim(maturity_code))) )),
       M06_temp = convert( float, sum(temp line amount * sign( patindex(
"M06", rtrim(maturity_code))) )),

       M12      = convert( float, sum(line_amount * sign( patindex(
"M12", rtrim(maturity_code))) )),
       M12_temp = convert( float, sum(temp_line_amount * sign( patindex(
"M12", rtrim(maturity_code))) )),

       Y01      = convert( float, sum(line_amount * sign( patindex(
"Y01", rtrim(maturity_code))) )),
       Y01 temp = convert( float, sum(temp line amount * sign( patindex(
"Y01", rtrim(maturity_code))) )),

       Y02      = convert( float, sum(line_amount * sign( patindex(
"Y02", rtrim(maturity_code))) )),
       Y02 temp = convert( float, sum(temp line amount * sign( patindex(
"Y02", rtrim(maturity_code))) )),

       Y03      = convert( float, sum(line_amount * sign( patindex(
"Y03", rtrim(maturity_code))) )),
       Y03 temp = convert( float, sum(temp line amount * sign( patindex(
"Y03", rtrim(maturity_code))) )),

       Y05      = convert( float, sum(line_amount * sign( patindex(
"Y05", rtrim(maturity_code))) )),
       Y05 temp = convert( float, sum(temp line amount * sign( patindex(
"Y05", rtrim(maturity_code))) )),

       Y07      = convert( float, sum(line_amount * sign( patindex(
"Y07", rtrim(maturity_code))) )),
       Y07 temp = convert( float, sum(temp line amount * sign( patindex(
"Y07", rtrim(maturity_code))) )),

       Y10      = convert( float, sum(line_amount * sign( patindex(
"Y10", rtrim(maturity_code))) )),
       Y10_temp = convert( float, sum(temp line amount * sign( patindex(
"Y10", rtrim(maturity_code))) )),
       Y30      = convert( float, sum(line_amount * sign( patindex(
"Y30", rtrim(maturity_code))) )),
       Y30_temp = convert( float, sum(temp_line_amount * sign( patindex(
"Y30", rtrim(maturity_code))) )),
       PE       = convert( float, sum(line_amount * sign( patindex(
"PE", rtrim(maturity_code))) )),
       PE temp = convert( float, sum(temp line amount * sign( patindex(
"PE", rtrim(maturity_code))) )),
       max_expiry_date = max(temp_expiry_date)

from   lines     l
group  by l.client_code,
          l.credit_code,
          l.LB_entity_code,
          l.client_role
```

This view merges the rows of the lines table and formats them as they are shown in CWS with ascending time periods in columns instead of rows.

WPI Credit Project Meeting

Thursday, 27 October 2005: 10:30 – 11:45

Purpose: Discussion on Client Workstation (CWS), EBS and Reuters email/data formats.

Present: Leesan Wong, Mahvish Ayoob, Tom McConnon, William Hays, Amy Jackson

Notes:

The main issue is the disjunction of credit line limits between the internal Lehman Brothers Credit Work Station (CWS) system and Reuters, EBS (New York), EBS (London), and EBS (Tokyo). The EBS London information is here, while to receive the EBS information from Tokyo and New York, it needs to be requested from the respective contacts.

A prerequisite to solving this problem is receiving the current data from EBS and Reuters sent on a regular basis, such as, weekly. To do this, Leesan will contact EBS and we will contact Robert Goldsmith and Brendan Murphy, two Lehman market data department employees. In order for the system under develop to be successful and efficient; the incoming data from EBS locations and Reuters needs to be in a consistent format sent on a consistent basis. Currently, EBS does not have a consistent format for their data. On the other hand, Reuters has the ability to send data in a consistent format. EBS London data is only accessible on the trading level floor.

There are three separate problems that have been understood pertaining to this project:

1. A unified display of all relative line limits for a particular counterparty.
2. A process for resolving counterparty groupings.
3. A process for identifying new counterparties (whose credit line has not yet been analyzed).

Before implementing anything with CWS, we need to develop a process for mapping EBS/Reuters dealing codes and long names with CWS client codes and client names for counterparties.

We should use 'ABN AMRO' as an example testing counterparty.

Line Limits Problem (Problem #1)

This check is done by checking the following equation:

Minimum (DSL1, DSL2, DSL3, DSL4, DSL5) <= Reuters + EBS_NY + EBS_LON + EBS_TOK

Where the DSL numbers are the daily spot limits for a particular counterparty with a legal entity of 'LBI' and a line type of 'FOREX.' Lehman Brothers Inc. (LBI) is where all trades for Reuters and

EBS are booked. Also, the Reuters and EBS numbers are the corresponding limit allocation numbers for the counterparty.

Counterparty Groupings Problem (Problem #2)

The counterparty parent groupings within CWS do not necessarily match those within the different EBS location system and Reuters.

Follow-up

- Ask Leesan for master source for groupings.
- Check with Scott Chang about how to integrate the system under development into the current CRM web page.
- Write up exactly what it is that we need from EBS/Reuters (formatting)
- Contact Robert Goldsmith and Brendan Murphy to discuss receiving Reuters data and possibly EBS data on an ongoing basis. These contacts will also discuss the current data downloading process.

WPI Credit Project Meeting

Friday, 28 October 2005: 11:30 – 12:00

Purpose: Discussion how FX limits are derived.

Present: Thomas McConnon, Jocelyn Girard, Mahvish Ayoob, William Hays, and Amy Jackson

Notes:

- Counterparties' FX lines are set based on credit rating (internal) and equity.
- CWS is a global LB system showing an analysis of counterparties.
- The analyst is concerned with not only the family appetites, but also the appetite of a particular party.
- FX limits are notional limits not MPE limits.
- Settlement lines, spot lines, and forward lines.
    - Settlement lines are amounts settled throughout all trades, the total amount of money out the door.
- There is a risk of the money coming in given the money has gone out (settlement exposure).
- *Net settle*: difference between FX rates.
- *CLS settle*: depends on the member and currency.
- EPS only has spot trading limits, 3 different amounts due to 3 different locations.
- Super group: spot limit for EMG and non EMG, this total equals the total apetite.
- In CWS (Non EMG), Group: FX, Legal entity: LBI Line: Spot.
- In Reuters, limit is generally for spot tading. Yes/No for forward trading. Yes relies a FX traders to go into the system to check how far to trade forward.
- As limits tear down, they decrease over time.
- Forward-shows limits forward (LBI FX, month and year columns in CWS)
- The FX desk sees limits in a different system that reads CWS limits, FX maria, FX ak?, CSS, same notional limits.
- LBI: plain FX (US)
- LBCC: FX options (US)
- LBFF: Doesn't include equity derivative (US)
- LBIE: (UK)
- LBJ: (Japan)
- Front desk only trades with non-regulated entities (LBFF, LBCC, LBFF).
- CSA: Credit Support Annex: pledge agreement
- Equity Masters: Shows if an ISDA Master Agreements are in place or not.

WPI Credit Project Meeting

Monday, 31 October 2005: 10:30 – 11:00

Purpose: Project update and discussion

Present: Leesan Wong, Mahvish Ayoob, William Hays, Amy Jackson

Notes:

- CWS only has information for legal entities, whereas EBS and Reuters contain both legal entity and branch information.
- Branch groupings may be different between EBS London, EBS NY, and EBS Tokyo. For the present task, we should focus on just EBS London, Reuters, and CWS data and investigate the other two EBS locations later.
- The branch information is important and there needs to be some sort of log.
- Remaining line limit allocation does not need to be tracked.

Next Steps

- In the near future we should sit down with Mahvish and go over some examples and ideas.
- We should also contact Jocelyn Girard about how the credit limits are produced (assigning different percents to assets of the company).

WPI Credit Project Meeting

Tuesday, 01 November 2005: 14:00 – 14:30

Purpose: Discussion on interface implementation with existing CRM web page

Present: Scott Chang (via telephone), William Hays, Amy Jackson

Notes:

- We should suggest to Leesan Wong that she notify the New York office about this project so not to duplicate work.
- Check about the ESM ID that is stored in the CWS system regarding counterparties.
    - Ask: Robert Chu or Tom McConnon
- There is a pre-approved limit project existing.
    - Lisa Raine is a NY Credit IT and the one doing this project.
    - Project may involve branch groupings.

CRM web page

- Start work with Weblogic
    1. Start at testing phase
    2. Then to staging in Q & A
    3. Production

Next Steps

- Ask Leesan Wong about notifying New York about this project.
- Ask Lisa Raine about branch groupings project.
- Ask Bappa Roy about Weblogic setup
- Ask Ford Tan about ESM ID

WPI Credit Project Meeting

Monday, 07 November 2005: 12:00 – 13:00

Purpose: Discussion on FX (foreign exchange) trading

Present: Christian Moeller, William Hays, Amy Jackson

Notes:

- Limit setting is based on trading.
- Tom next (tomorrow next): tomorrow's price settled the next day.
- Big players in FX trading tend to be commercial banks.
- Lehman Brothers is a client-driven investment bank - Lehman performs an FX trade if a client wants it, to facilitate customer trading.
- In Lehman, FX uses only notional limits. The plus side of this is it is easy to understand. The down side is volatility (currency fluctuations).
- Split limits: FX and FX EMG (emerging markets). FX EMG is more volatile.
- Think of credit risk from an opposite point-of-view as investing. If there is a profit of, say, 1million dollars, then there is an increase in credit exposure. From the credit point of view, it is better if the company is losing money.

Settlement Risk

Settlement risk is a form of credit risk that happens at the settlement of a transaction. In FX trading, settlement risk occurs because of the time difference between when exchange transactions are made. When a bank in one country transfers money in one currency to another bank in another country with another currency, the initial transaction is made from the initiating bank, during their trading hours. The return transaction is made in the second bank's trading hours. This difference in trading hours, or time lapse between transactions, is the reason a settlement risk occurs. Settlement risk is a risk that one of the banks will default during the time elapse (money goes out but doesn't come in).

Settlement risk came about after the failure of the German bank, Herstatt Bank (June 26, 1974). Transactions between banks with the longest difference in time zones have the greatest settlement risk. Transactions between banks in the same time zone have the lowest settlement risk. Large limits signify good visibility of the bank's probability to default.

Notional Limits

Notional limits are based on notional amounts (quantity of the underlier).

DSL: Daily Settlement Limit (1-5 days)

Spot Limits

Forward Limits (1-30 years)

Notional limits are hard to calculate risk.

Forward Limits

Forward limits are set based on forward risk (there are 2 risk factors in forward trades):

1. In the future, when the trade takes place there will be settlement risk.
2. Then there is a risk the market moves and the counterparty defaults. The risk the market moves is price risk (or market risk) calculated with counterparty exposure (FX model).

(Research Forward Hedging)

CLS

Continuous Linked Settlement (http://www.cls-group.com/) allows foreign exchange transactions to be settled within the same day, eliminating settlement risk and leaving only price risk as the risk for the trade. CLS is a "clearing house" for trades. There is currently a list of 15 eligible currencies. A few banks started CLS and now there are hundreds of third-party members.

Spot Limits

Spot limits are equal to the settlement limits if there are no forwards at the time of settlement. If there are forward contracts ending on the same day as a spot trade, then the settlement risk is higher. Spot transactions are settled in 2 days.

Swaps

Swaps are usually an overnight market. Swaps occur when two counterparties exchange cash flows, usually associated with interest rates. It is also related to FX trading. Swaps are different than forwards because there is an exchange at the beginning and end of the trade. This is the reason a counterparty would participate in a swap, you need money to buy. CLS cannot aid swaps in the final transaction – not the initial one. Swaps are usually done for funding purposes.

Options

An option gives the buyer the right to perform a transaction with the seller according to specified terms (later in FX "life"). When an option is traded, a premium is initially paid (sometimes over time). To the seller, there is no risk after the buyer has paid this premium; for this reason the seller then has no credit risk. The buyer, on the other hand, has higher risk. There are call and put options for the buyer and seller. Call options give the buyer the right to buy an underlier. Put options give the buyer the right to sell an underlier. An option can be exercised before a set expiry date. The highest risk, in terms of credit risk, is when Lehman is the buyer (as shown below in the grey).

LB	call	put
buy		
sell		

Delta

Delta captures the relationship between the current price and the option price. Delta of a call option is positive; Delta for a put option is negative. Limit setting can't just be based on notional limits. It also has to be based on the delta adjustment (talk to Leo to find out about this).

WPI Credit Project Meeting

Monday, 07 November 2005: 10:30 – 11:00

Purpose: Discussion about Limit Scenarios

Present: Mahvish Ayoob, William Hays, Amy Jackson

Notes:

Scenarios (from previous notes, listed for reference):

1. The counterparty and its limit is only listed in CWS.
2. The counterparty and its limits are in CWS and EBS Global but not Reuters.
3. The counterparty and its limits are in CWS and Reuters but not in EBS Global.
4. The counterparty is only found in CWS and EBS NY.
5. The counterparty is found in CWS, Reuters, EBS London, and EBS Tokyo, but not in EBS NY.
6. The counterparty is only found in CWS, EBS London and EBS Tokyo.
7. The counterparty name is found in all sources except EBS London.
8. In CWS the counterparty is listed in two different locations, but in EBS and Reuters it is just listed as one group.

- If a counterparty has a limit in CWS and no other sources (1) then it should be flagged to be researched to see if the analysts want to allocate it in Reuters and EBS.
- If a counterparty has a limit in CWS and Reuters but not EBS Global or a limit in CWS and EBS Global but not Reuters (2, 3) then it more than likely means it hasn't been looked into for that source. This also should be flagged and researched.
- If a counterparty is only in CWS and some EBS location (4, 6) then it might be because it not traded globally. This also should be flagged and researched.
- If a counterparty is not found in all the locations (5, 7) it may be because no trading is done near the location it is not listed in and then therefore there is no appetite at this location. This also should be flagged and researched.
- In the case where the grouping is different between all the different sources (8), then an investigation needs to be done to make sure all the groups listed are indeed legal entities. This is done by first looking on the client's website to look up subsidiaries and branches. Then second contacting the analyst. This also should be flagged and researched.

[Generalized]

- If a limit is missing in the database, it should be flagged and then adjusted after research.
- Any limit changes go through Leesan.
- When reporting errors/scenarios, a scenario code should be associated with the counterparty situation.
- There should also be a flag that asks which duplicate limit entry to delete when there is a name change.

WPI Credit Project Meeting

Friday, 11 November 2005: 13:30 – 14:15

Purpose: Database Demo

Present: Leesan Wong, Mahvish Ayoob, William Hays, Amy Jackson

Notes:

- Make sure everything will drill-down to the dealing code.
- Some multiple names in Reuters or EBS may map to one legal name in CWS (ABN AMRO) and some multiple names may be duplicated and need to be deleted.
- Amendments to the counterparty screen:
    - Label the units of the limits.
    - Freeze the headings.
    - Change the label "Legal Name" to "Group Name" because some of the names listed in Reuters and EBS are not always legal names.
    - Define that they are FX LBI Spot Limits.
- Should label the sub groups "Sub Entities" not "Branch" because sometimes the name is the legal name and sometimes it is a branch name.
- We requested a list of reports that they would like. Known wanted reports include:
    - Clients with more than one limit in one source
    - Names not in CWS

References

Bouchet, Clark, and Groslambert. Country Risk Assessment: A Guide to Global Investment Strategy. West Sussex, England: John Wiley & Sons Ltd, 2003.

"Credit Risk". Risk Glossary. 2005. Contingency Analysis. 10 Dec. 2005 <http://www.riskglossary.com/link/credit_risk.htm>.

Hull, John C. Options, Futures, and Other Derivatives. Prentice Hall, 2002.

Jorion, Philippe. Value at Risk: The Benchmark for Controlling Market Risk. Blacklick, OH, USA: McGraw-Hill Professional Book Group, 2000.

Lehman Brothers. 2005. Lehman Brothers Inc. 10 Dec. 2005 <http://www.lehman.com/who/history/index.htm>.

"Moody's". Wikipedia. 1 Nov. 2005. Wikimedia Foundation, Inc. 10 Dec. 2005 <http://en.wikipedia.org/wiki/Moody%27s>.

"Standard & Poor's". Wikipedia. 1 Nov. 2005. Wikimedia Foundation, Inc. 10 Dec. 2005 <http://en.wikipedia.org/wiki/Standard_and_Poors>.

Wilkin, Sam. Country and Political Risk: Practical Insights for Global Finance. London, England: Risk Books, a Division of Incisive Financial Publishing Ltd, 2004.