Developer's Guide to FOREX 2.0

Data Extractions

J. Yu
Material Group Operational Research

DRDC CORA TN 2007–05
May 2007

Defence R&D Canada
Centre for Operational Research and Analysis

Material Group Operational Research
Assistant Deputy Minister (Material)

National Defence Défense nationale

Canada

Developer's Guide to FOREX 2.0

Data Extractions

J. Yu
Material Group Operational Research

DRDC – Centre for Operational Research and Analysis

Technical Note

DRDC CORA TN 2007–05

May 2007

Author

J. Yu

Approved for release by

P.E. Desmier
Director, Material Group Operational Research

The information contained herein has been derived and determined through best practice and adherence to the highest levels of ethical, scientific and engineering investigative principles. The reported results, their interpretation, and any opinions expressed therein, remain those of the authors and do not represent, or otherwise reflect, any official opinion or position of DND or the Government of Canada.

Abstract

FOREX (FOReign EXchange) is an application used to quantify foreign exchange risk. In preparing the input data, the prototype model required a significant amount of manual work to be done with Microsoft Excel© spreadsheets in conjunction with Visual Basic macro scripts. In so doing, errors could easily have been made due to the amount of time spent manually updating fields on large datasets.

This second version attempts to minimize the amount of work to accomplish the same task. It is designed to be a net application which utilizes a SQL (Structured Query Language) database in order to quickly and efficiently sort through data and provides the output the user desires.

Résumé

FOREX (FOReign EXchange) est une application qui sert à quantifier le risque lié aux échanges avec l'étranger. Pour préparer les données d'entrée, le modèle prototype exigeait une importante somme de travail manuelle à l'aide des feuilles de calcul électronique Microsoft Excel combinées avec des macros de Visual Basic. Ce faisant, il est fort possible que des erreurs se soient produites en raison de la quantité de temps consacrée à la mise à jour manuelle des champs dans de vastes ensembles de données.

Cette seconde version vise à minimiser la somme de travail requise pour accomplir la même tâche. Elle se veut une application réseau qui utilise une base de données SQL (langage d'interrogation structuré) pour trier de manière rapide et efficace les données et fournir les sorties requises par l'utilisateur.

This page intentionally left blank.

Table of contents

Figures

Acknowledgements

Sincere appreciation is extended to: Dr. P.E. Desmier for his original work in developing the FOREX model as well as providing insight into the final outcome of this project; Mr. Kenneth K.B. Hiseler, DG Fin Ops, for providing all raw Financial and Managerial Accounting Systems (FMAS) data.

1 Introduction

1.1 Background

FOREX was originally developed under MS Excel with VBA Macros to calculate Foreign Exchange Risk for the ADM(Mat) National Procurement (NP) and Capital accounts for the three currencies: USD (U.S. Dollar), EUR (Euro) and GBP (U.K. Pound Sterling). Through this method, a lot of work had to be done manually. The requirements for FOREX version 2.0 was to limit the amount of manual work needed while still accomplishing the same results with accessibility via the web.

1.2 Scope

In this Technical Note, the different modules used to retrieve and store relevant data will be discussed together with additional debugging modules that may prove to be helpful.

Following the introduction, section 2 describes the software used in creating FOREX version 2.0.

Section 3 gives a full analysis of the programming modules used to achieve the data retrieval and archival process. Each module is broken down into subsections where specific variables or constructs are explained in further detail.

Section 4 covers code that is used for debugging.

Section 5 provides an explanation of the current state of the project and also outlines any relevant problems.

Section 6 provides a brief overview of what still needs to be developed for the project to come to fruition.

2 Software

This section is dedicated to explaining the software involved in this project.

2.1 Data Retrieval and Archival

The software used in the project for the purpose of retrieving and storing data is limited to PHP and Apache.

2.1.1 Apache

Apache is an open-source HTTP Web server software which supports languages such as PHP. The developer responsible for designing this portion of the product has experience with Apache and Windows Internet Information Services (IIS) web servers and has chosen Apache as the hosting web server, however this can be changed to IIS if required.

2.1.2 PHP

Hypertext Preprocessor, otherwise known as PHP, is a programming language used in developing dynamic web pages. Due to the dynamic nature of the product as well as the need for logical expression evaluation, PHP was a logical choice. PHP files have the extension *.php*, and behave similarly to HTML files but with the benefits of declaring variables, objects, functions, etc.

Other languages such as C and Java were considered, but adding another software package would only require additional maintenance.

PHP Extension and Application Repository, otherwise known as PEAR, is a distribution system for PHP code. PEAR provides an online repository of code developed and reviewed by the PHP community. The modules uploaded to PEAR must adhere to the coding standards outlined in [1] to be listed for the public.

Client URL, known as cURL, is one of the modules which was developed for the PEAR repository. cURL is originally a command line tool for transferring files with URL syntax, but has also been ported over to be used in PHP scripts. The use of cURL in this section of the project is limited to contacting the Bank of Canada website and retrieving the stored data. This functionality is provided in the module *curlfun2.php*.

File_archive is another PEAR module which is used to unzip files using PHP code. It is dependent on another PEAR module, MIME_type, and is used in unzipfun.php.

SQLite is a relational database system contained in a small C library which comes packaged with PHP5. SQLite is responsible for all the SQL transactions in this portion of the FOREX project but this may change in the future (see Section 5). SQLite was chosen instead of MySQL (another free relational database system) for a variety of reasons. SQLite is fast

and the databases are contained in simple files due to database administration not being necessary. SQLite also implements a large subset of the SQL 92 standard [2] and was deemed sufficient in handling the queries required for this portion of the project.

3 Data Retrieval and Archival

The main programs which call all modules responsible for storing/accessing data are *main.php* and *intro.php*. In the sections to follow, a brief description of each module is given along with the current state of progress and suggested improvements. Figure 1 displays a graphical representation of what occurs in the main program.

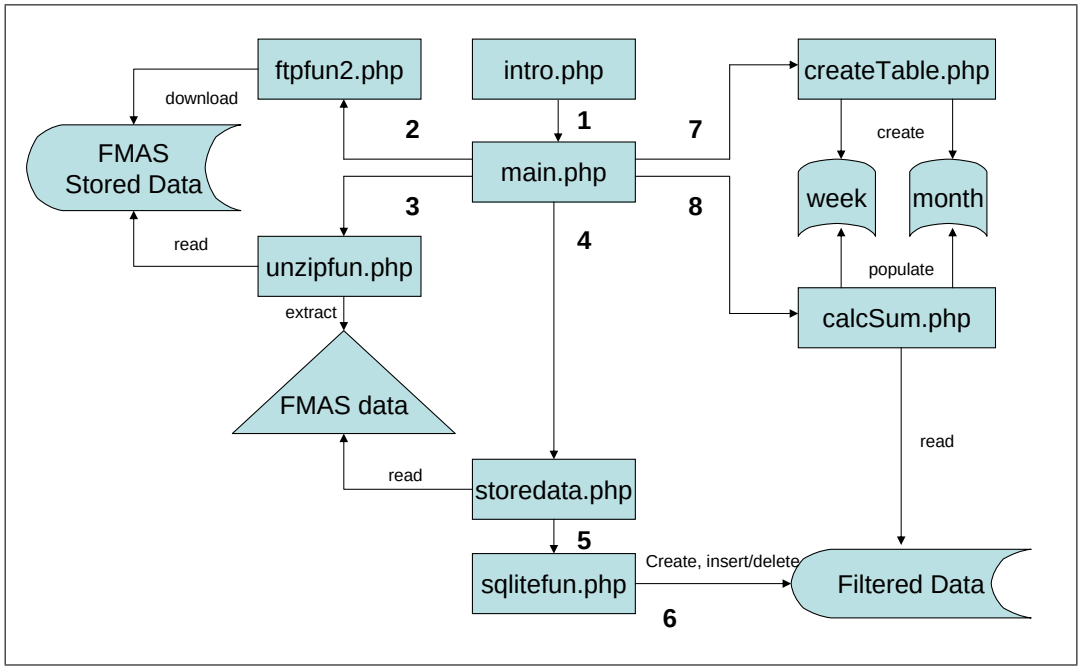

Figure 1: *Process flow of data retrieval/archival*

3.1 intro.php

The primary function of this module is to accept input from the user. The user must first select a fiscal year (provided via a drop-down menu (see Figure 2)) in which to analyze and store the data into database format. This module is the only one that the requires user interaction in this portion of the project. The only variable that is required is the drop down menu option.

In the current release of FOREX v2.0, the field *fiscal_year* is used to determine which fiscal year to analyze. The information is passed to *main.php* and used in subsequent modules. Due to the static nature of the data coupled with the availability, it is suggested that this process be automated by scheduling a task (Windows) or running a Cron job (*nix)[1]. This is a relatively minor improvement and is thus not a priority.

[1] Cron allows Unix users to execute commands and scripts automatically at a specified date and time

Figure 2: *User Interface Screen*

3.2 main.php

The function of *main.php* is to call any modules responsible for retrieving and storing necessary data. Main.php calls *ftpfun2.php* to download a zipped input file. The file is then unzipped utilizing the function provided by *unzipfun.php*. The file *storedata.php* will then be called to store relevant data into a SQL database. Sum tables are then initialized and updated via *createTables.php and calcSum.php* respectively. The following is a code snippit from main.php which depicts the process described above.

```
echo "Contacting FTP Server...";
//call ftp
echo "Contacted <br \>";
echo "Unzipping...";
//call unzip
echo "Unzipped <br \>";
```

```
echo "Storing Data...";
store_data(\year);
echo "Stored Data <br \>";
echo "Creating SumTables <br \>";
createSumTables($year);
echo "Created Tables <br \>";
// Calculate Weeky & Monthly Sums for $year
echo "Calculating Weeky & Monthly Sums for $year";
calcSum($year);
echo "Calculated";
```

In the current release of *main.php*, the files *ftpfun2.php* and *unzipfun.php* are **not** called. These files work with PHP 5.1.4 and above, with the appropriate PEAR (PHP Extension and Application Repository) files installed on the computer. All the necessary files should be installed on the local development machine however, *ftpfun2.php* will only run on a D-WAN connected machine as it retrieves data from the intranet. As such, these files have been manually copied to the development machine.

This file should require no further modifications with the exception of the integration of *ftpfun2.php and unzipfun.php* into *main.php*. A further improvement is to perform this task once annually via a scheduled task on Windows or the *nix equivalent (cron job).

3.3 ftpfun2.php

This file contains a function *getftpfile($year)* which will download the input file from the FMAS FTP server. The following code indicates the variables that are specific to this particular download. The variable $local_file designates the location where the file will be downloaded to whereas $server_file refers to the location of the file on the $ftp_server.

```
$local_file = "./extracts/cc7a".substr($year, 2,2).".zip";
$server_file = './pub/sapextract/BFY_'.substr($year, 2,2).
 '_Files/cc7a'.substr($year, 2,2).'.zip';
$ftp_server = "fmas-ftp.desc.mil.ca";
```

The current release can be considered the final release in regards to its use in this project.

3.4 unzipfun.php

This file contains a function *fileunzip(&$year)* which will unzip the downloaded input file. The following code indicates the source and destination path of the zipped file and it's contents.

```
$src = File_Archive::read("./extracts/cc7a".substr($year, 2,2).".zip"),
$dest = File_Archive::toFiles("./extracts")
```

The current release can be considered the final release in regards to its use in this project.

3.5 construct3.php

This file contains two classes which are used in the files to follow. The object *dataObject* contains information on each object in the annual database. This object exists to pass information to the databases. The object *dateObject* stores the number of the week and month for each object. These numbers are explained further in Section 3.8. Figure 3 illustrates both objects and the elements it holds. The variables are self-explanatory as they are based on their names with the exception of $np_cap and $delFlag. The variable $np_cap holds either the value 'NP' or 'CAP' for designating either the national procurement or capital account respectively. The variable $delFlag is a recent addition and is used to flag variables which will be deleted (due to SQLite's limitations of modifying 2 tables at once). This variable may be removed in the future if this functionality is not required.

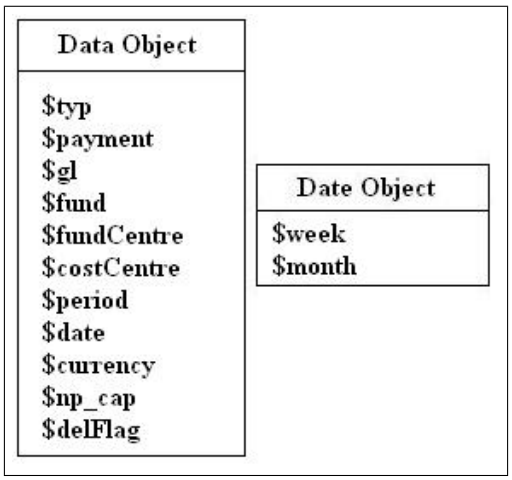

Figure 3: *Data and Date Objects defined in construct.php*

3.6 storedata.php

The file *storedata.php* contains a module *store_data* which reads input from a text file and creates an annual database of items which fit certain criteria. Data is stored in sqlite3 files located in the subdirectory "C:/program files/apache group/apache2". Each annual database is labeled YEAR.sqlite3, where YEAR is the end of that specific fiscal year. Figure 4 illustrates what a sample database would contain. The columns (from left to right) are as follows: Row id, type, payment, GL, fund, fund centre, cost centre, period, date, currency, np or capital, month number, week number, flag for delete. The current release still requires a slight modification and is not complete. At the end of the file, the function *deleteRel($year)* is commented out because a working copy does not exist yet. This function is currently being tested on the file*temp2.php* but again, does not work properly at the moment.

0	SA	-0.01	8560	C503	2184BB		12	20100331	USD	CAP	53	12
1	SA	-0.1	8560	C503	2184BB		12	20100331	USD	CAP	53	12

Figure 4: *Sample database contents*

There is possibly a more efficient method of reading, filtering, and storing the input but was left for a later release. Valid items in the current release include KR/SA type items with funds C113, C503 or C513 and a currency type of USD, CAD, GBP or EUR.

3.7 sqlitefun2.php

This file creates and updates the annual database with relevant data objects. The function *checkAndInsert($year, $obj)* will only insert relevant data objects into a table called allObject(see reference [3]). Relevant data that has a negative payment property will be stored in a table called negObject. The data in negObject should be compared to allObject to remove any positive values where applicable (see reference [3]). This comparison functionality is not fully implemented yet.

It should be noted that the check to see if the database exists does not work. If the file does already exist it will start appending to the current database.

3.8 datetest.php

This file determines the week and month number associated with a given date within a fiscal year (April to March). Each week begins on a Sunday and ends on a Saturday. Due to this convention, there may be more or less than 52 weeks in any given year. As an example, the date Saturday, April 1st 2006 is given a week number and month number of 1, whereas Sunday, April 2nd 2006 has a week number of 2 and month number of 1. Also note that Sunday, April 30th, 2006 has the same week number as Monday, May 1st, 2006, but different month numbers.

The current release can be considered the final release in regards to its use in this project. The week number is determined based on the above algorithm, however it was deemed that the month number should simply follow the value of the period field (from the input file).

3.9 createTable.php

This file contains a function *createSumTables($year)*. The purpose of this module is to create and initialize two tables to store the data into weekly and monthly divisions. Due to the static nature of the data used in this project, it was decided that storing this data was a more suitable solution than performing a summation every time the program was run. In this way, a simple SELECT statement would be sufficient for data retrieval. The two database files can be found in “C:/program files/apache group/apache2”.

The current release can be considered the final release in regards to its use in this project. One possible improvement to this file would be to store the *cur_type and nc_type* (currency

10	360474.5	EUR	CAP
11	479893.3	EUR	CAP
12	2272432	EUR	CAP
1	312937.1	GBP	NPR
2	2165770	GBP	NPR
3	1385059	GBP	NPR
4	1510238	GBP	NPR
5	2549387	GBP	NPR
6	2468263	GBP	NPR
7	4726721	GBP	NPR
8	6162795	GBP	NPR
9	1228866	GBP	NPR

Figure 5: *Sample summation database contents*

type and NP and capital respectively) arrays into text files that would be read. This would provide users with an easy method of adding more currency and account types without modifying the existing code. This however would also require changes in the filtering process in *storedata.php*.

3.10 calcSum.php and sumoutput.php

These files will search the sum of all item payments per account type and currency in every week and month of the year. It will then update the weekly and monthly tables with these values. Figure 5 illustrates the contents of these databases.

3.11 curlfun3.php

This module gathers foreign exchange rate data from the Bank of Canada website [4] and stores it into a database called 'BOC.sqlite'. It utilizes cURL (client Uniform Resource Locator) functions and requires cURL to be enabled on PHP to work.

The current release is now integrated with the main program (called by *main.php*) and requires no user interaction at all.

4 Debug Code

This section is reserved for debug code that may prove to be useful. In the current version, the only debug code is *output.php*.

4.1 output.php

This code allows the developer to view the state of any of the databases created in easy to read CSV (Comma Separated Values) files. To use this module, simply modify the $db_file and $sqlSelectVal variables and run the script. If the output is greater than 50,000 lines, the module will automatically split the file into 50,000 chunks.

The program in its current release has no user interface and requires hardcoding to account for any changes. In the following code snippit from *output.php*, the 3 variables that tells the program which database to output is shown. The variable $year provides the year to output (which is used in lines 3 and 4). The variable $db_file provides the location of file which stores the database. Note that $db_file is related to the variable $sqlSelectVal such that line 2 should refer to lines 8 or 9; line 3 refers to line 10; line 4 refers to line 11; line 5 refers to line 12.

```
[1] $year = 2005;  // Input year here
[2] $db_file = "/../$year.sqlite3";
[3] $db_file = "/../MonthlySum$year.sqlite3";
[4] $db_file = "/../WeeklySum$year.sqlite3";
[5] $db_file = "/../BOC.sqlite3";
[6]
[7] //SQL Statements
[8] $sqlSelectVal = 'SELECT * FROM allObject'; //Positive Database
[9] //$sqlSelectVal = 'SELECT * FROM negObject';  //Negative Database
[10] $sqlSelectVal = 'SELECT * FROM MonthlyTotal';
[11] //$sqlSelectVal = 'SELECT * FROM WeeklyTotal';
[12] $sqlSelectVal = 'SELECT * FROM BoC';
```

This module could be modified to receive input from the user through textboxes. This will reduce interaction with the code and result in less chance of unnecessary modifications. Additionally, the column names could be displayed on each new filename as the first row for increased clarity.

5 Current Implementation

In regards to the project as a whole, the data retrieval and archival process is near completion but still requires some work. Currently the data in the summation tables are incorrect due to the fact that data is not being deleted as expected (see Section [3]). The valid negative data is stored into the database negObject in YEAR.sqlite3 whereas the positive data is stored into the database allObject in the same file. Follow-on development is expected to perform the necessary comparisons and deletions such that the summation tables will output the correct amount.

Due to limitations in SQLite's parser, the current implementation was unable to provide the required functionality in a quick and simple method. Given more time, a workaround could possibly be achieved, however it may be simpler to re-implement the database structure using MySQL or another relational database engine. MySQL is recommended because it is commonly used with PHP and functions similarly to SQLite.

All php files can be found in "C:/program files/apache group/apache2/htdocs" whereas all sqlite3 database files are located in "C:/program files/apache group/apache2/". Due to project still being in its relative infancy, none of the files are hosted on a web server. As such, the files can only be run on the development machine (localhost). The User Interface can be called by opening a web browser (Firefox or Internet Explorer) and typing *localhost/intro.php* or simply *localhost* if the php.ini file is modified correctly. The file *Output.php* can be called similarly by typing *localhost/output.php* in the address bar of a web browser.

6 Future Development

Data manipulation is the next major process for this project. Originally, the data was manually entered into MS Excel and utilized the SOLVER add-on to calculate the Maximum Likelihood Estimation (MLE). In FOREX v. 2.0, the data is stored in SQL databases and, as such, requires another method to calculate the MLE. It has been suggested to utilize a Downhill Simplex Model for this calculation. An explanation of this model as well as examples written in C are provided in section 10.4 of reference [5].

A user interface option to select a date range for calculations (for both Bank of Canada data and FMAS data) should also be provided. This option will allow the user to perform the estimation of foreign exchange rate exposure to the Department of National Defence and could be used as a validity check with the original version of the FOREX model.

Output should always be available in *.csv* or *.xls* formats for ease of use for the user. Although examples are given in C, developers should not be restricted to programming in C and are encouraged to utilize whichever programming language they choose.

References

1. The PEAR Group (2007). PEAR - PHP Extension and Application Repository - Coding Standards (Online). http://pear.php.net/manual/en/standards.php (April 2007).

2. DevZone (2004). SQLite - Introduction (Online). ttp://devzone.zend.com/node/view/id/760#Heading4 (April 2007).

3. Desmier, P.E. (2007). Estimating Foreign Exchange Exposure in the Department of National Defence. (DRDC CORA Technical Report TR 2006–23). Defence R&D Canada, Centre for Operational Research and Analysis. Materiel Group Operational Research.

4. Bank of Canada (2006). Rates and Statistics: Exchange Rates (Online). http://www.bankofcanada.ca/en/rates/exchange.html (April 2007).

5. Central Michigan University Office of the Comptroller (2006). Accounting Services (Online). http://www.nrbook.com/a/bookcpdf.php.

List of symbols/abbreviations/acronyms/initialisms

*nix	Unix or Linux Machine
CAD	Canadian Dollar
cURL	Client URL
EUR	Euro
FOREX	FOReign EXchange
FTP	File Transfer Protocol
FMAS	Financial and Managerial Accounting Systems
GBP	Great Britain Pound Sterling
KR	Vendor Invoice (German)
IIS	Internet Information Services
PEAR	PHP Extension and Application Repository
PHP	Hypertext Preprocessor
SA	G/L Account (German)
SQL	Structured Query Language
MLE	Maximum Likelihood Estimation
URL	Uniform Resource Locator
USD	United States Dollar

1630-1 (DMGOR)

May 2007

Distribution List

Developer's Guide to FOREX 2.0 – Data Extractions

References: A. J. Yu, *Developers Guide to FOREX 2.0 – Data Extractions*, DRDC CORA TN 2007–05, May 2007 (enclosed).
B. P.E. Desmier, *Estimating Foreign Exchange Exposure in the Department of National Defence*, DRDC CORA TR 2006–23, January 2007.

1. Reference B documented the theory and application of a model that was built on forecasting expenditures for the ADM(Mat) National Procurement (NP) and Capital (equipment) accounts and the time-varying volatilities of foreign currency returns. Reference A documents a first step in making the model operational by describing the data retrieval, filtering and archival mechanisms required for the two accounts and the Bank of Canada exchange rates prior to further analysis.

2. Questions or comments are welcome and should be addressed to Dr. P.E. Desmier at (613) 996-8440, CSN 846-8440 or by email at *desmier.pe@forces.gc.ca*. Electronic copies of this report are also available upon request from *Repsys.R@forces.gc.ca*.

P.J. Comeau
Section Head Joint & Strategic Analysis
for Director General Centre Operational Research & Analysis

Enclosures: 1

Distribution List

DRDC CORA Library (2)
DRDKIM (2)
Spares (3)

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

DRDC – Centre for Operational Research and Analysis
NDHQ, 101 Col By Drive, Ottawa ON K1A 0K2

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

Developer's Guide to FOREX 2.0

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Yu, J.

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc).	6b. NO. OF REFS (total cited in document)
May 2007	24	5

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

Technical Note

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address).

DRDC – Centre for Operational Research and Analysis
NDHQ, 101 Col By Drive, Ottawa ON K1A 0K2

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Specify whether project or grant).	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written).

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique.)	10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC CORA TN 2007–05	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution beyond the audience specified in (11) is possible, a wider announcement audience may be selected).

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

FOREX (FOReign EXchange) is an application used to quantify foreign exchange risk. In preparing the input data, the prototype model required a significant amount of manual work to be done with Microsoft Excel© spreadsheets in conjunction with Visual Basic macro scripts. In so doing, errors could easily have been made due to the amount of time spent manually updating fields on large datasets.

This second version attempts to minimize the amount of work to accomplish the same task. It is designed to be a net application which utilizes a SQL (Structured Query Language) database in order to quickly and efficiently sort through data and provides the output the user desires.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

FOREX
PHP
PEAR
cURL
SQLite

DRDC CORA

www.drdc-rddc.gc.ca