UPS Capital

Insurance Management System (IMS)

System Overview

April 2014

Version 2.0

Table of Contents

Table of Figures

1. Executive Overview

The Insurance Management System (IMS) is part of the overall Trade Risk Services (TRS) architecture. IMS is a Windows® application consisting of four modules:

- Billing Management
- Invoice Management
- Revenue Management
- Security Module

IMS is intended to replace all functionalities currently provided by *Velocity*, a 3rd party software application. IMS has been developed in house and is wholly owned by UPS Capital.

General Insurance Management Process

Following is a general business process for the insurance management at TRS.

- An insurance policy is created in the CDEP web application (CDEP is an acronym for Communication, Delegation, and Empowerment Program).
- Additional parameters and detailed billing information are added and saved in the Billing Management module.
- Shipping information and package level detail (PLD) are obtained from the UPS Common Billing Layout (CBL) billing file for the shippers that are covered under FLEX policies and the CDV program.
- An invoice is generated based upon shipping information and policy coverage.
- Payments are received and applied to UPS Capital accounts in the Revenue Management module of IMS
- Reports are generated based upon data contained in the IMS Billing and Invoice modules.
- All permissions for access to each module are managed with the Security module of IMS.

The figure below indicates the general flow of processes at IMS.

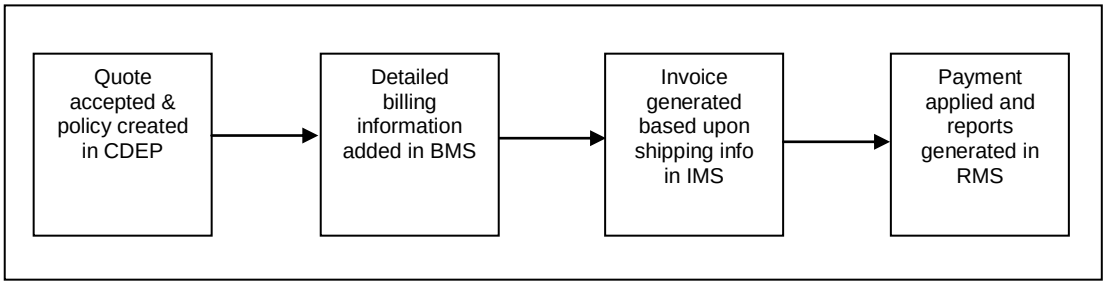


Figure 1 - 1 "Insurance Policy Process"

1.1. IMS Business Process

1. BDMs create quotes via RFQ in SalesFunnel.
2. BDMs accept quotes on shippers' acceptance.
3. TRS Operations creates a policy in CDEP.
4. CDEP feeds policy data into BMS.
5. Operations users enter additional policy data into BMS which creates billing parameters for shippers.
6. PLD with shipping information is downloaded from the UPS billing files.
7. Invoice data is fed into the IMS (Invoice Management System).
8. An invoice is generated.
9. Invoices are sent out.
10. Payments are received and entered into RMS.

1.2. Application Programming Information

IMS is a Windows® client server application that runs on a Windows® XP® platform. All of the files are stored in a server named GAALPSVR0329. Execution files and client files are all kept in IMSCLIENT.

IMSSERVER communicates between the IMSCLIENT and the GAALPSVR0329 server. The data is stored in the SQL 2005 database server on GAALPSVR0329.

VB.NET 2005 is the development environment, and VB.NET is the programming language.

Remoting Technology is used to communicate between client and server. The Single Call method is used to serve the data.

TCP/IP network protocol with port 8080 is used to communicate with client and server. Also the transport link uses binary format to transfer the objects between client and server.

IMS implements remoting technology for the following reasons. .NET remoting provides an abstract approach to inter-process communication that separates the remotable object from a specific client or server application domain and from a specific mechanism of communication. As a result, it is flexible and easily customizable.

The communication protocol can be replaced with another protocol, or one serialization format can be replaced with another without recompiling the client or the server. In addition, the remoting system assumes no particular application model. It can communicate from a Web application, a console application, a Windows Service – from almost anything that needs to be used. Remoting servers can also be any type of application domain. Any application can host remoting objects and provide its services to any client on its computer or network.

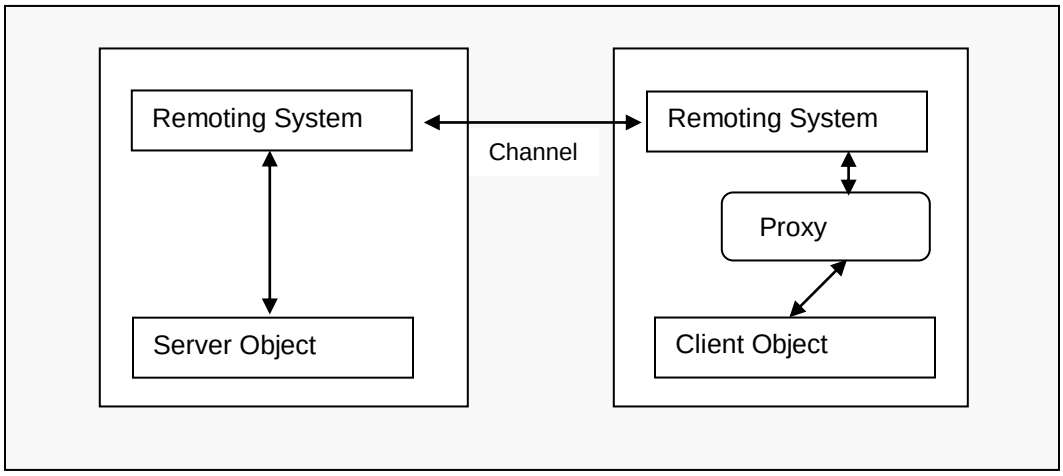


Figure 1 - 2 "Remoting System"

1.3. IMS Architecture

The IMS application has a three-tiered architecture that consists of multiple clients, a single server for both application and multiple databases. Each module within the three-tiered architecture contains the following layers:

- Presentation layer – consisting of UI forms
- Business logic layer – consisting of functions
- Data access layer – which accesses the database tables

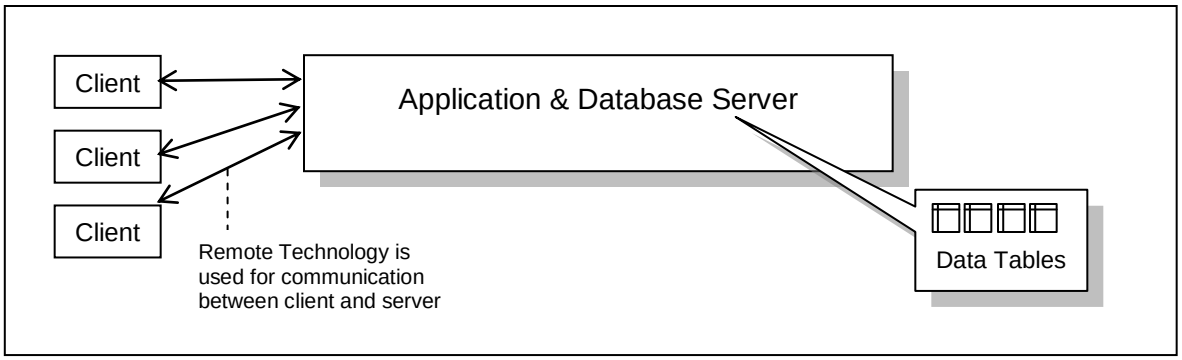


Figure 1 - 3 "Current IMS Architecture in one server environment"

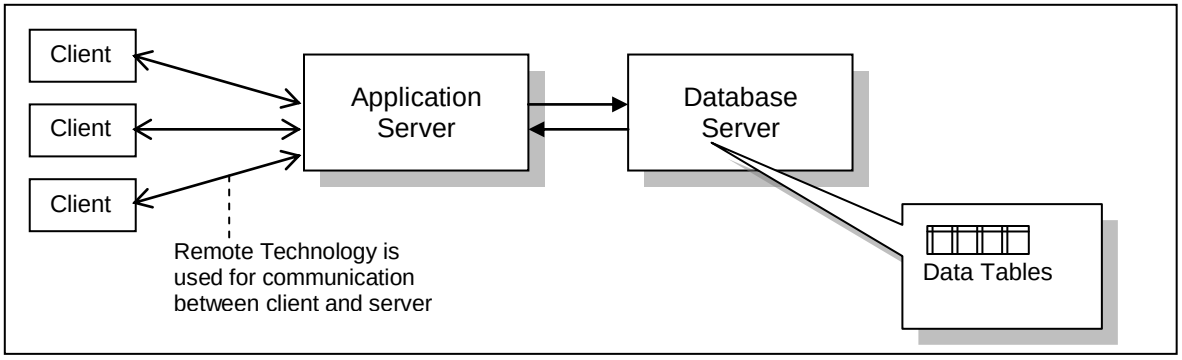


Figure 1 - 4 "IMS Architecture in two servers (application & database) environment"

Multiple clients access the server to perform tasks or return information. The application can access multiple tables in the Database server, if necessary, and then return data to the client. The clients never access the databases directly. The application has been tested in both one server (both application and databases on a single server) and two server (application and databases on two different servers) environments.

Within the Insurance Management System there are four different modules, each with different classes of data. The four modules are:

- Billing Management Module
- Invoice Management Module
- Revenue Management Module
- Security Module

All four modules follow the same basic process:

1. UI makes a request.
2. Request is sent to the server.
   - Server processes request
   - Server gets data from database
3. The database server returns the data in the appropriate dataset.
4. The server returns data to the UI.

1.4. Design Patterns

IMS application has been designed using a structural design pattern called ADAPTER. The design pattern below has been carried over in all the IMS modules wherever applicable.

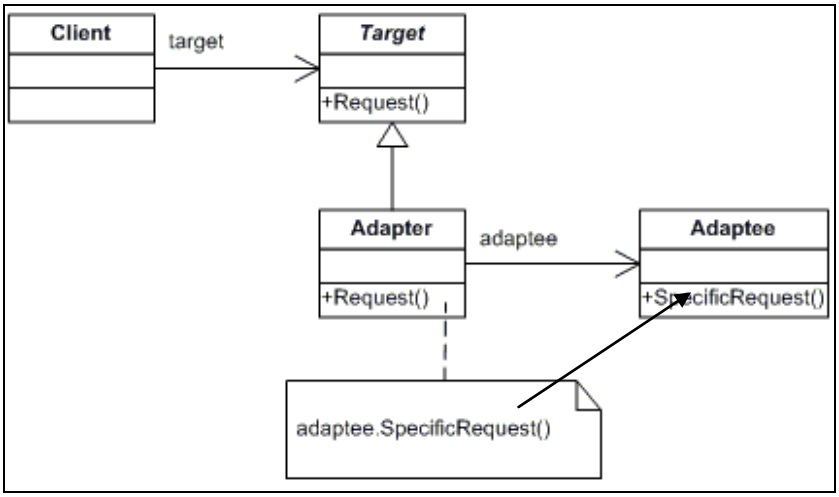


Figure 1 - 5 "UML Class Diagram"

The classes and objects participating in this pattern are:

- Client: IMS Client
- Target: Datasets
- Adapter: IMS server class which has manager and data access class
- Adaptee specificRequest(): Common data access layer
- Adaptee: IMS databases

1.5. IMS Development Environment

Each module within IMS is composed of three classes:

- Executable
- Dataset
- Server

Additionally, two other classes are used:

- Common Utils – performs common tasks for billing, invoice, and revenue functions
- UPSC Controls – developed in house

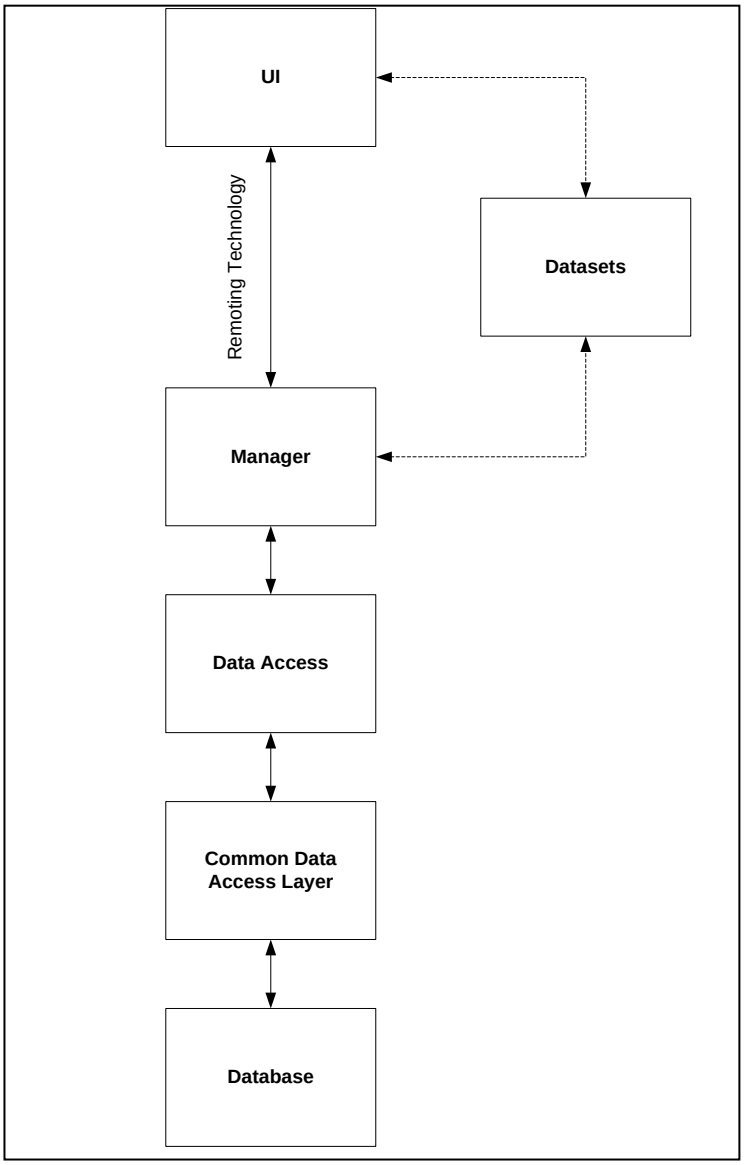


Figure 1 - 6 "IMS Development Environment Structure"

- The UI class is responsible for the UI forms. The UI sends a request to the server using Remoting Technology. Common Utils checks for security whenever the Main Menu form is loaded.
- Dataset – an extra dataset is used to transport data between the executable and the Manager. This dataset is used only when it is required, depending upon which function is called.
- The server is composed of two sub-classes:
    - Manager layer – contains the business logic and determines whether any calculations are necessary
    - Data Access layer – passes the data
- A Common Data access layer exists between the server and the database. The common data access layer is where the dataset is filled with data from the database. It consists of a common set of functions that are being used to obtain data from the database.
- The Database contains the following:
    - Stored procedures and functions
    - Data tables

All UI forms follow the Microsoft standard for code types:

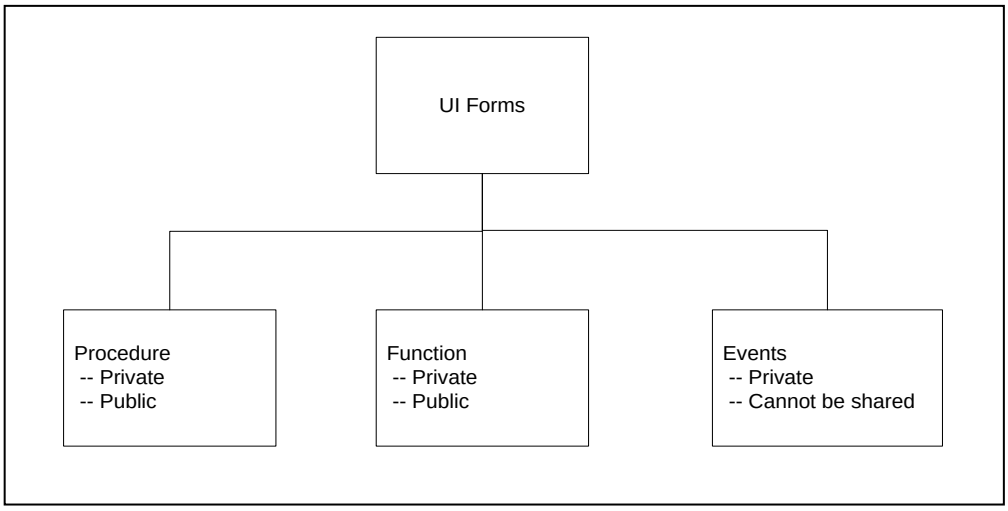


Figure 1 - 7 "IMS Forms Structure"

1.6. Programming Style and Naming Conventions

This section specifies the naming conventions used for the files in the development environment.

1.6.1. UI Naming Convention

The common naming convention for all of the UI forms is to use the prefix: **`frm`**

Examples:

- **frm**Billing
- **frm**EditBilling

1.6.2. Dataset Naming Convention

Each dataset file has the suffix: **`db`**

Examples:

- Policy**db**
- Address**db**

1.6.3. Server Naming Convention

Each part of the Server class has a different naming convention:

- Manager class has the suffix **mgr**
- Data Access class has the suffix **da**

Examples:

- Billing**mgr**
- Billing**da**

1.6.4. Variable Naming Conventions

All variables are declared with a meaningful name.

Examples:

- PolicyNo – the Policy Number variable
- BillingType – the Billing Type variable

1.6.5. Function Naming Conventions

All functions are given meaningful names.

Examples:

- CheckPolicyNumber
- EditPolicyNumber

1.7. Data Tables – Billing Management System

The figure below is an Entity Relationship for the **Billing Management System.**

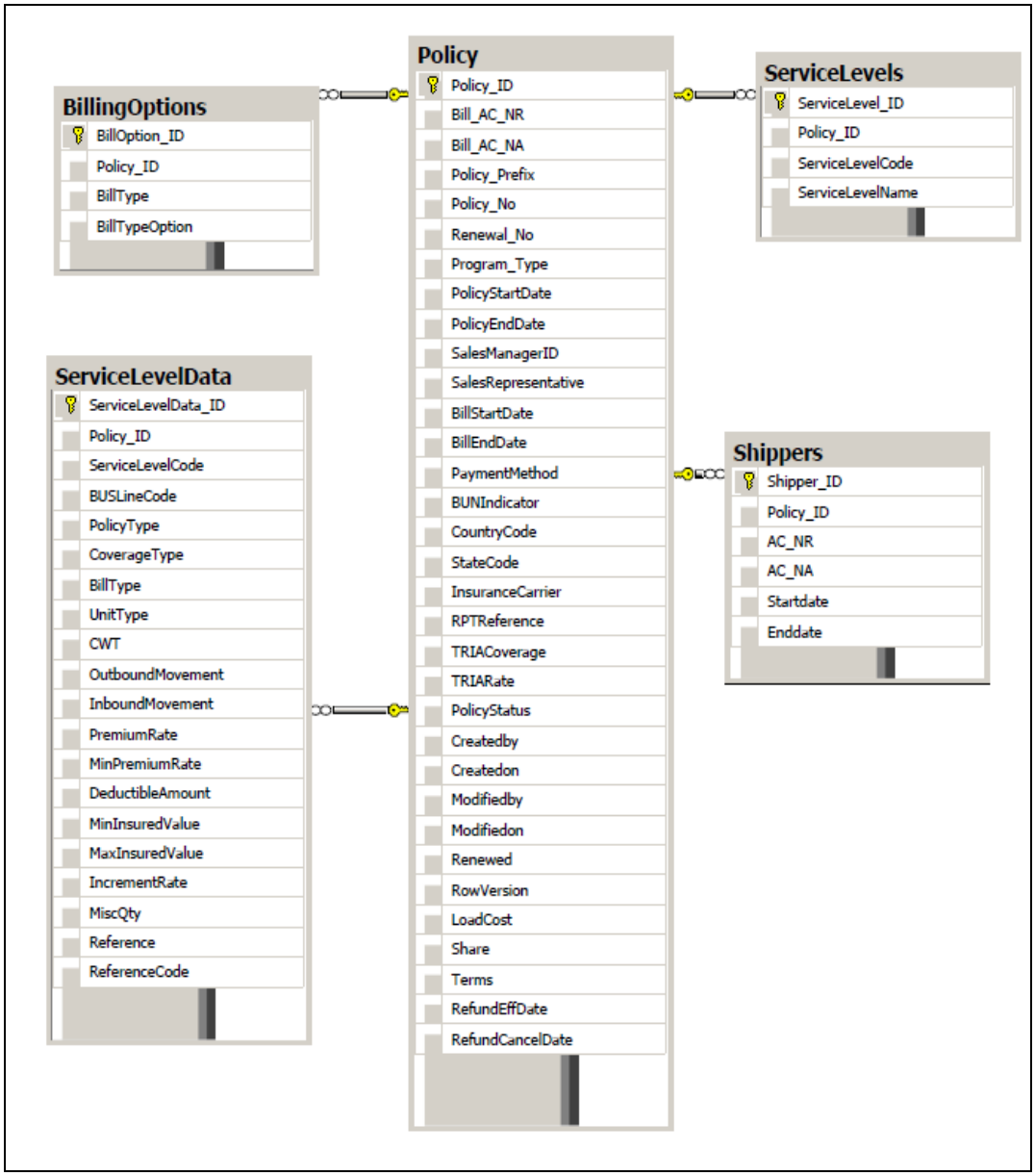


Figure 1 - 8 "Billing Management Data Tables"

The figure below contains the tables within the Policy Data server.

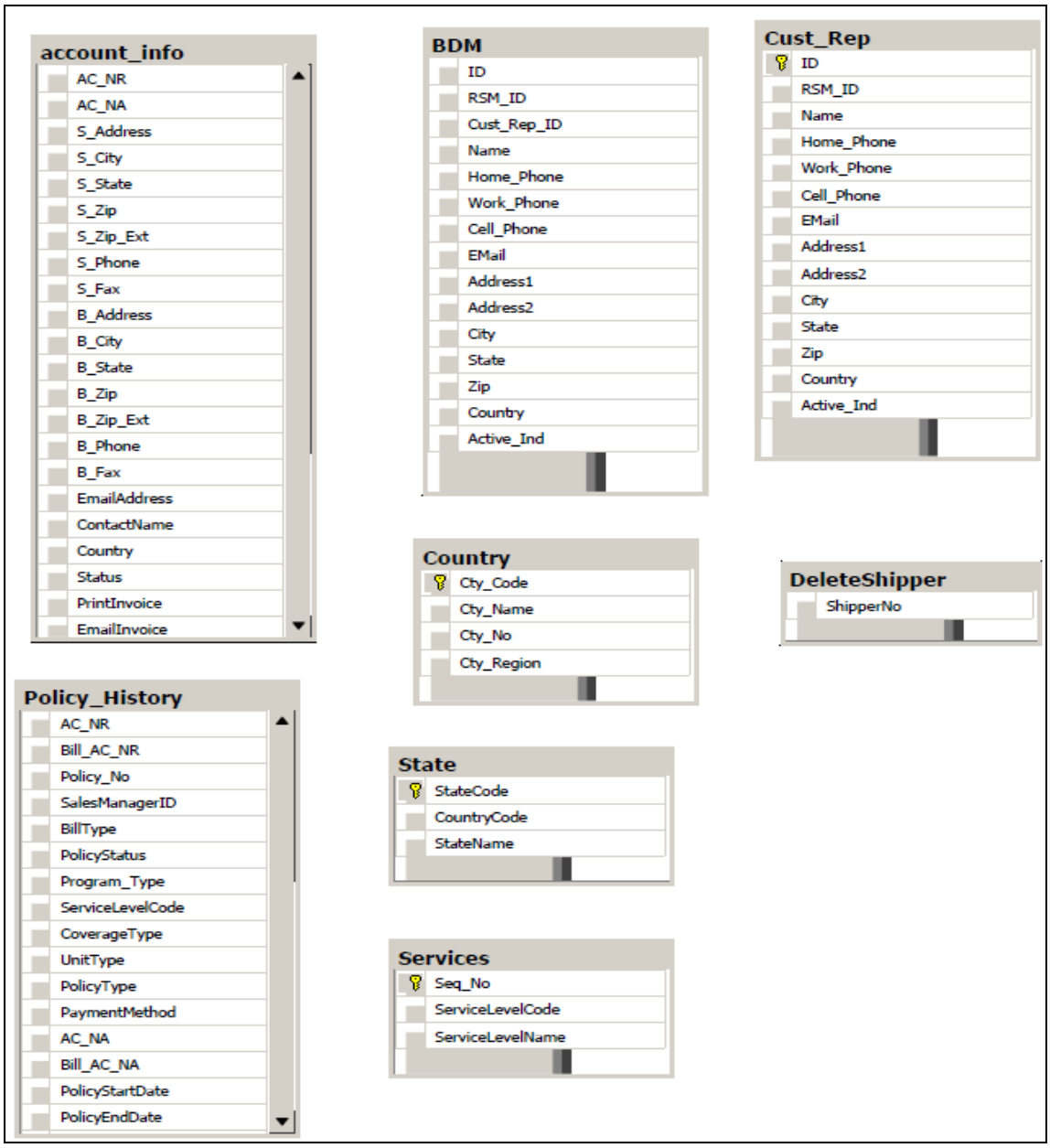


Figure 1 - 9 "Policy Data Server"

1.8. Data Tables – Invoice Management System

The figure below indicates the tables used by the Invoice Management System module.

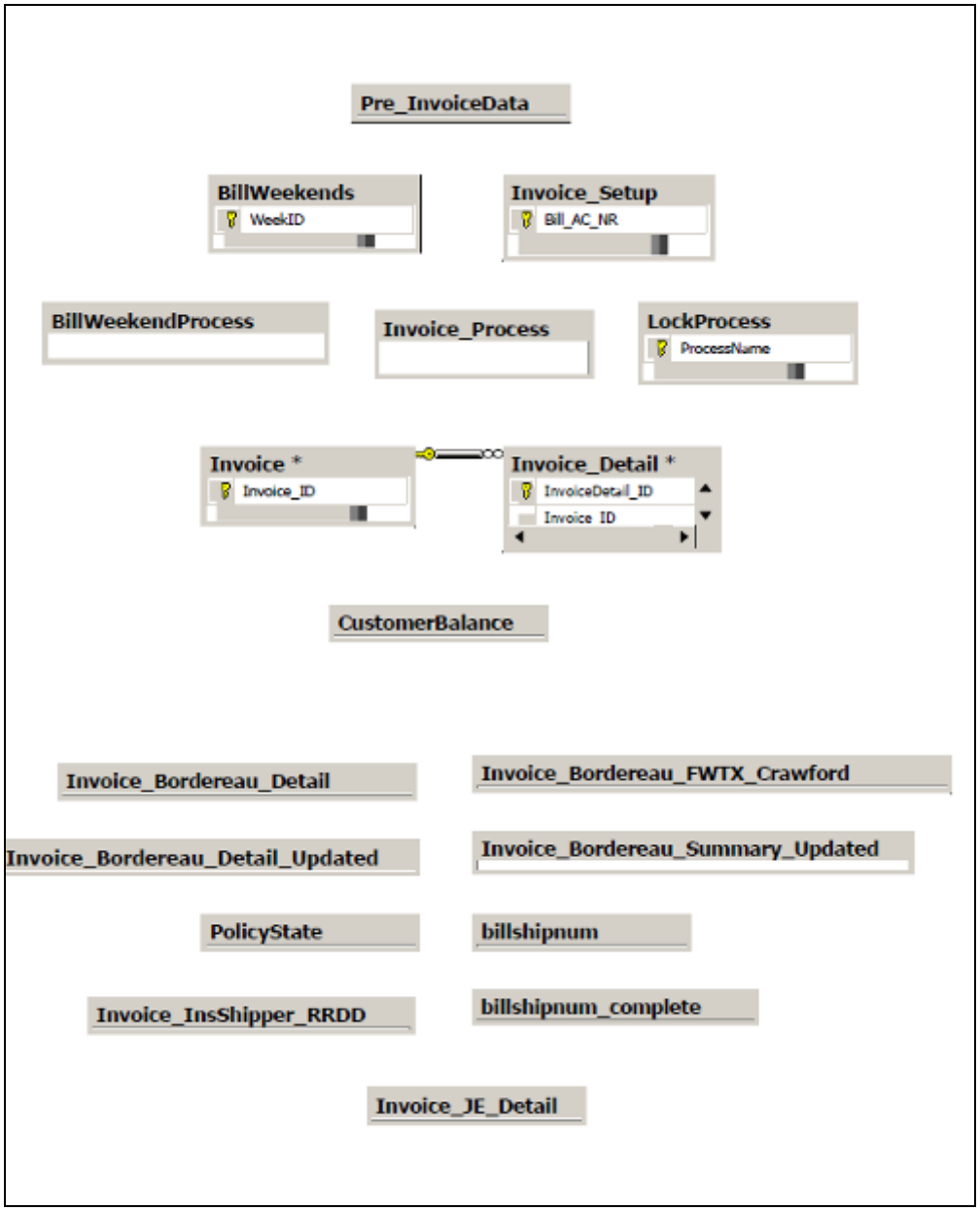


Figure 1 - 10 "Invoice Management System Data Tables"

1.9. Data Tables – Revenue Management System

The figures below indicate the tables used by the Revenue Management System module.

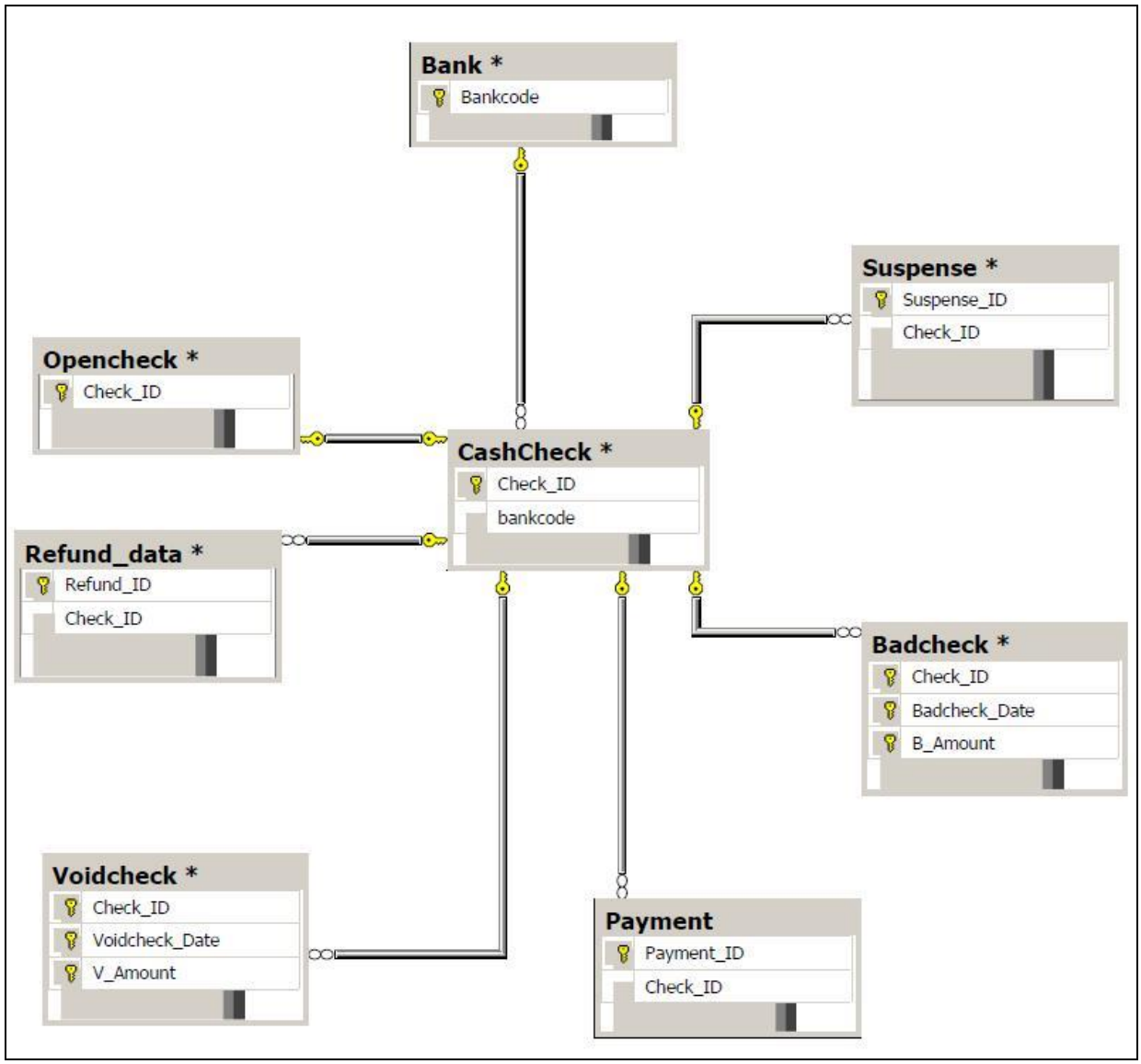


Figure 1 - 11 "Revenue Management System Data Tables"

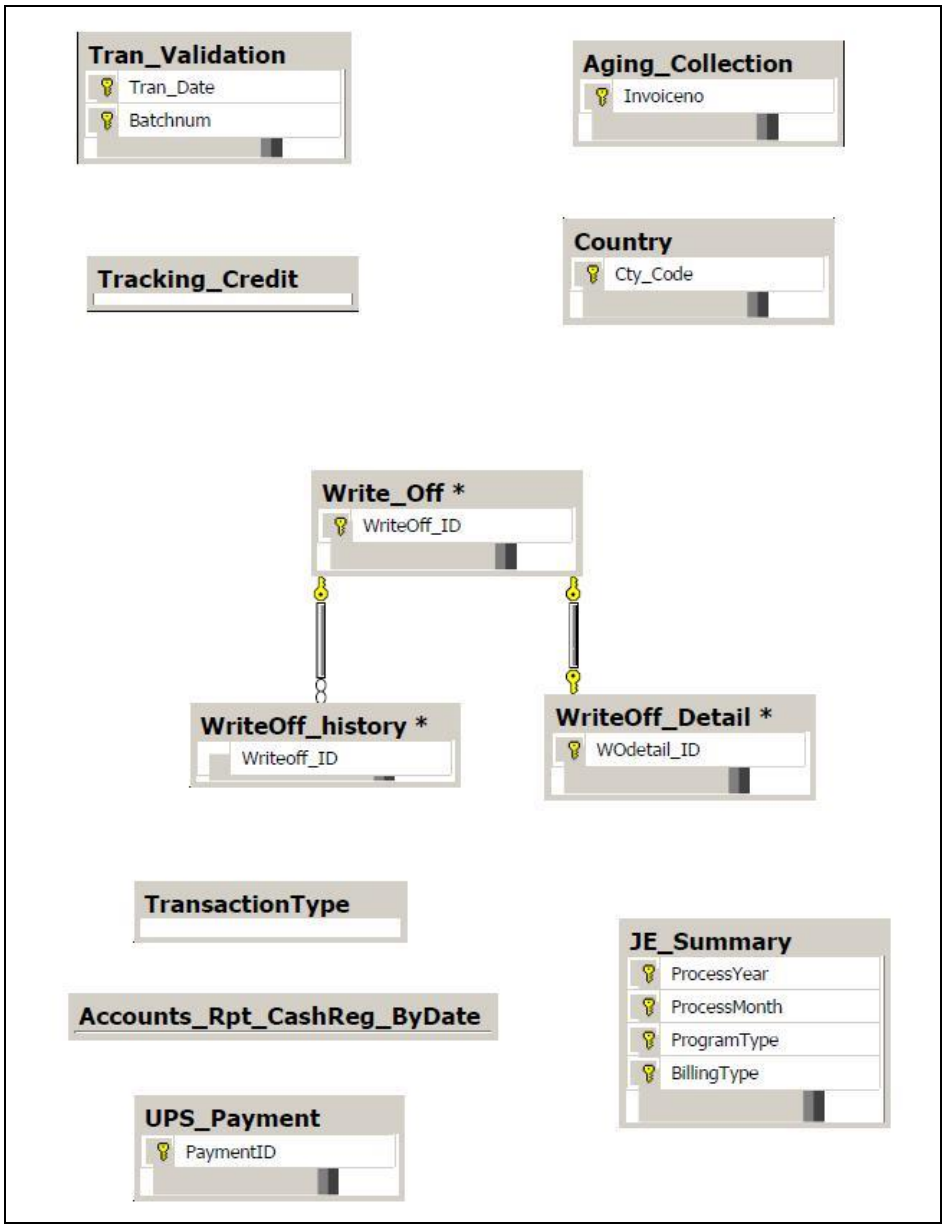


Figure 1 - 12 "Revenue Management System Data Tables"

2. Billing Management System

2.1. Overview

Initial customer information is fed into the Billing Management System after Business Development Managers have negotiated a contract and entered RFQ information into CDEP. Operations users enter additional shipping and billing information into BMS. The purpose of entering additional data is to create and manage a policy. Operations users create a billing record in BMS.

2.2. Major Functionality

The functions within the Billing Module are:

Main Function	Minor Function
Create	CDEP Policy Manual Policy
Edit	
Renew	Renew with NO Change Renew WITH Change

The functionality indicated in the table above applies to the following policy types:

- FLEX
- Canada
- UPS Stores
- Open Cargo

2.3. Stored Procedures

The following table contains a list and a description of the stored procedures used in the Billing functionality.

Stored Procedure	Description
Billing_CheckPolicyNo	Checks whether policy no exists or not.
Billing_GetBillShipperNo	Gets the Bill shipper information.
Billing_GetPolicyList	Gets the policy information.
Billing_GetPolicyRenewalList	Gets the policy for renewal.
Billing_GetShipperAddress	Gets Shipper Address.
Billing_GetShipperInfo	Gets the shipper information.
Billing_GetShipperName	Gets Shipper Name.
Billing_InsertorUpdateAddress	Gets Shipper Address.

Stored Procedure	Description
Billing_GetUPSStoreList	Gets the UPS Store list.
Billing_GetPolicyInfo	Gets the policy for renewal.
Billing_GetPolicyListFromChangeRequest	Gets the policy information.
Billing_GetShipperList	Gets the shipper information.
Billing_UpdateInvoiceSetup	Update Account_Info table with Invoice setup details.
Billing_GetStates	Gets the state information.
Billing_BulkPolicyRenewal	Gets the policy information.
Billing_GetPolicyDetails	Gets the policy information.
Billing_GetChangeofAddress	Gets the policy information.
Billing_GetInvoiceSetupDetails	Get Invoice Setup Details.
Billing_GetServiceLevels	Gets all the service levels.
Billing_BulkPolicyRenewal	Renew all the policy at once.
Billing_SetChangeRequestStatus	Set the change request status.
Billing_GetSalesManagerList	Gets the sales manager information.
Billing_GetCSRList	Gets the sales manager information.

3. Invoice Management System

3.1. Overview

The Invoice Management System provides functionality for importing invoice data, setting up invoices, creating invoices, and managing invoice delivery.

3.2. Major Functionality (Functions)

The major functions provided by the Invoice Management module are:

- Create invoice
- View invoice
- Print invoice

3.3. Stored Procedures

The following table contains a list and a description of the stored procedures used in the invoice functionality.

Stored Procedure	Description
Invoice_GetInvoiceForVerification	This procedure returns records page by page.
Invoice_GetInvoicesToEmail	Gets Invoice to email.
Invoice_GetJEInvoiceStatus	Get JE Invoice Status.
Invoice_GetInvoiceSetupDetails	This procedure returns records page by page.
Invoice_GetPolicyInfo	Gets Policy information for a shipper.
Invoice_GetShipperAddress	Gets Shipper Address.
Invoice_SearchInvoice	Search Invoices based on the condition provided.
Invoice_GetInvoiceTotal	Gets Invoice total.
Invoice_GetBillShipperNo	Gets Bill Shipper #.
Invoice_ImportInvoiceDetailData	Check billweekends exists or not.
Invoice_CreateManualInvoiceRecords	Create Records for Manual Invoice.
Invoice_UpdatePrintStatus	Updates Print status.
Invoice_CheckInvoicePrintStatus	This procedure checks the print status of invoice.
Invoice_GetBalanceDue	Gets Balance due total.
Invoice_UpdateInvoiceStatus	Updates Invoice status.
Invoice_UpdateBalanceDue	Update current Balance Due in Invoice

Stored Procedure	Description
	table.
Invoice_GetMaxWeekEndDate	Gets Max weekend date.
Invoice_GetAllBillWeekendDates	Gets bill weekend dates.
Invoice_GetPerviousBillWeekDate	Gets Previous Bill weekend date.
Invoice_GetShipperInvoiceInfo	Gets Shipper Invoice information.
Invoice_UpdateInvoiceDateForShipper	Updates Invoice date for a shipper.
Invoice_UpdateInvoiceDate	Updates Invoice date.
Invoice_GetInvoiceForPrinting	Get the Invoices that are Printed/Not Printed.
Invoice_GetNewInvoices	Check billweekends exists or not.
Invoice_CheckJEProcessRequiredOrNot	Close Month End.
Invoice_GetAllErrorsFortheWeek	Gets all the error records.
Invoice_GetEmailAddress	Gets Email Address.
Invoice_GetInvoiceSetupInfo	Gets Invoice setup information for a shipper #.
Invoice_CheckMonthClosedorNot	Checks whether month is closed or not.
Invoice_UpdateInvoiceProcess	Create a record in Invoice process table.
Invoice_GetMonthCloseStatus	Get Month Close Status.
Invoice_UpdateMonthCloseDates	Update Month Close Date.
Invoice_GetBillWeekEndStatus	Get the status of Billweekend.
Invoice_CheckPreInvoiceData	Check billweekends exists or not.
Invoice_UpdateBillWeekAvailableStatus	Updates the status of Billweekend avilability.
Invoice_GetAvailableBillWeekends	Get the billweekends that are ready to import.
Invoice_GetInvoice	Gets all invoices based on supplied parameters.
Invoice_GetBillWeekendDate	Get Billweekenddate for the supplied date.
Invoice_GetYear	Get Year.
Invoice_CheckBillWeekEndStatus	Checks whether month is closed or not.
Invoice_GetBillWeekEnds	Gets top 10 bill weekend date.
Invoice_MonthEndClose	Close Month End.
Invoice_ResetBillWeekendDateInvoiceStatus	Resets Bill Weekend date for the invoice status.

4. Revenue Management System

4.1. Overview

Once invoices are sent out and payments are received, the revenue is posted to cash accounts, suspense items are reconciled, and data is gathered via reports.

4.2. Major Functionality (Functions)

Major Functionality provided by the Revenue Management System:

- Posting Cash Payments
- Managing Reports
- Suspense Posting
- Check Processing
- Setting Month-End Close

4.3. Stored Procedures

The following table contains a list and a description of the stored procedures used in the Revenue Management functionality.

Stored Procedure	Description
Revenue_GetMaxWeekEndDate	Gets Max weekend date.
Revenue_GetAllBillWeekendDates	Gets bill weekend dates.

5. IMS Security

5.1. Overview

The IMS Security module provides user security for the entire IMS application.

5.2. Functionality

The IMS Security module allows the developer to set security access to all screens and menus for the entire Insurance Management System application. Access can be set at the screen level and then further defined by controls available within each screen. A developer can set up the security for one user and then copy the security profile to other users of the same type.

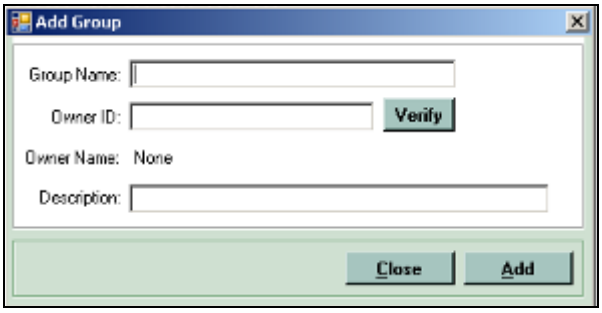
The Security module provides the following functionality:

- Add a new group
- Edit a group
- Add user types
- Assign roles to all user types
- View controls under user profiles
- View users who lock a process and unlock the process if necessary
  Note: Some functions are locked to new activity while key processes are running.

5.3. Security Screens

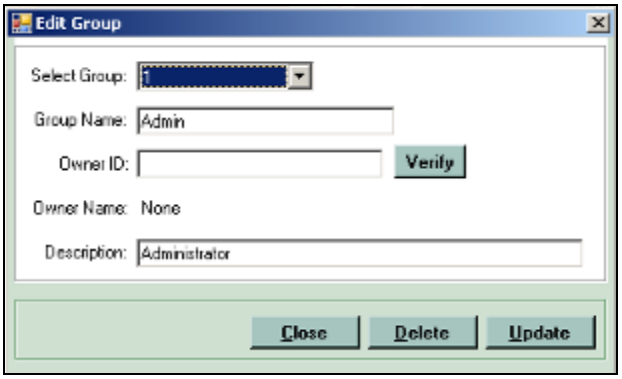
The screens on the following pages are used to update system security.

5.3.1. Add a Group



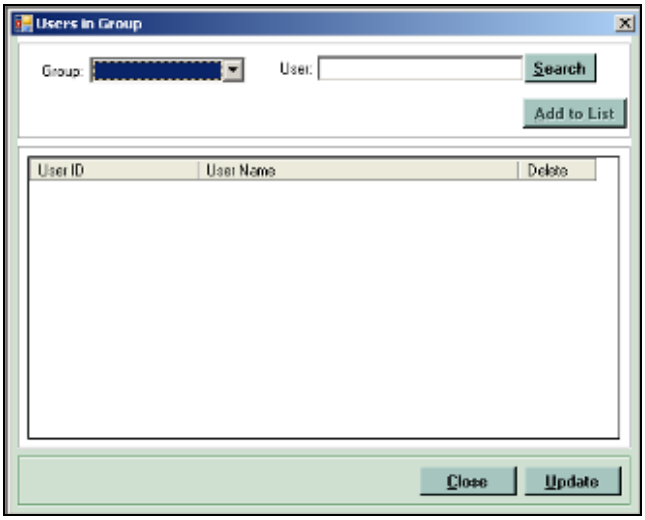
Menu Path	Security > Group > Add Group
Functionality	Use this screen to add users to a particular group in the security module. Groups of users have similar functionality.

5.3.2. Edit Group



Menu Path	Security > Group > Edit Group
Functionality	After you have set up a group, you can edit the group characteristics.

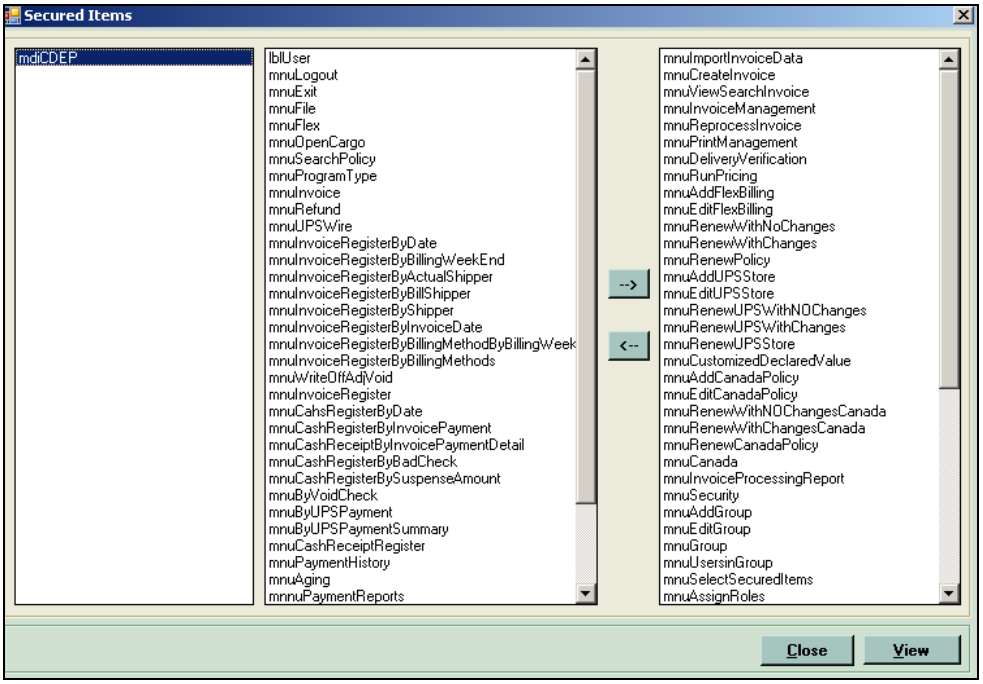
5.3.3. Users in Group



Menu Path Security > Users in Group

Functionality After you have set up a group, you can add users to it.

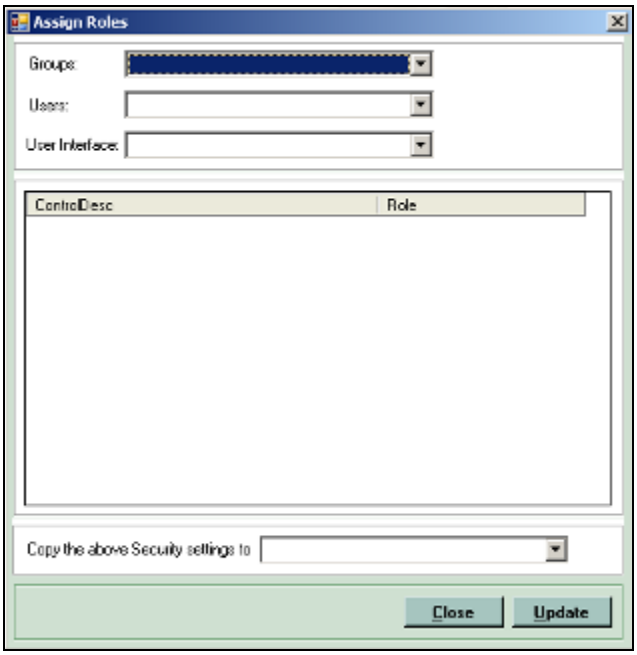
5.3.4. Secured Items



Menu Path Security > Select Secured Items

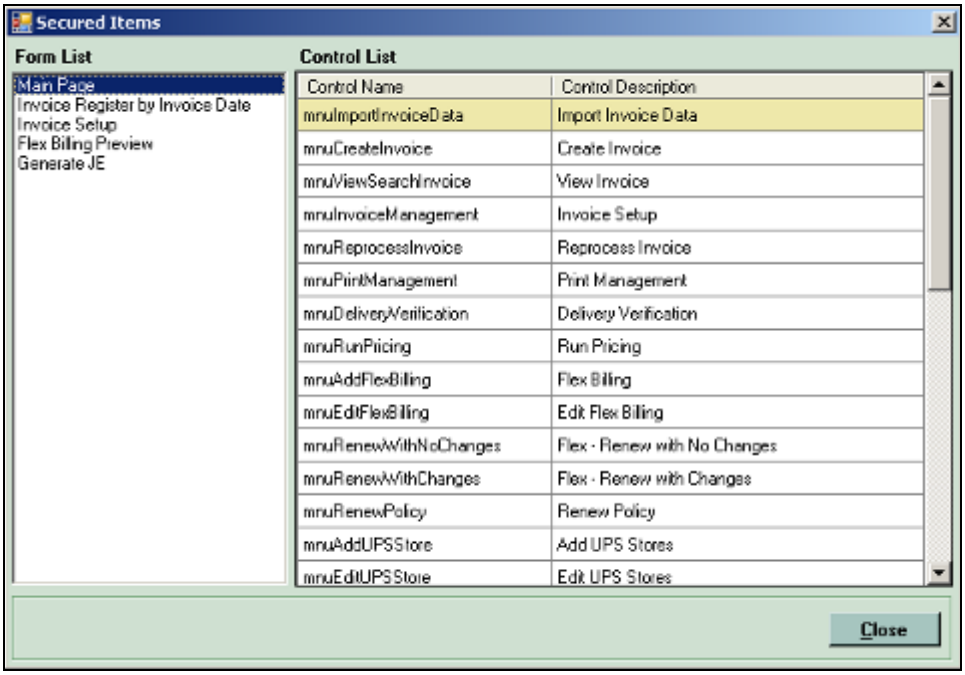
Functionality This screen allows you to view all controls available and add controls to a user group.

5.3.5. Assign Roles



Menu Path	Security > Assign Roles
Functinality	Once have added users to a particular group, you can assign roles and copy security settings.

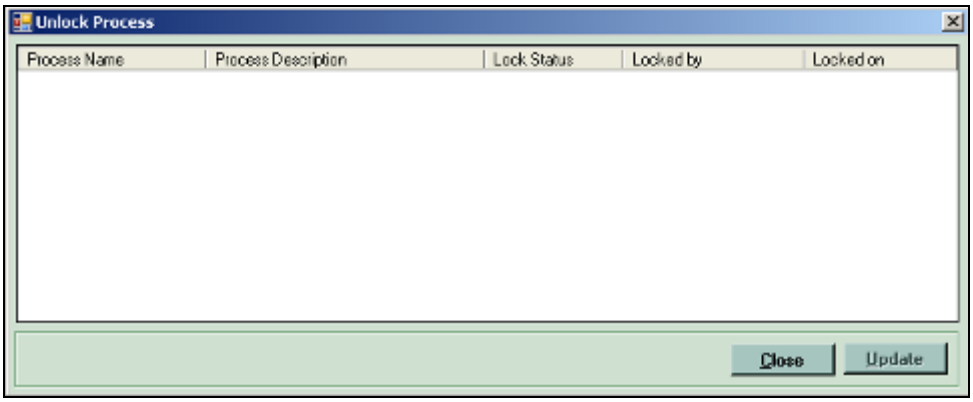
5.3.6. Secured Items



Menu Path Security > Display Secured Items

Functionality This screen displays a list of controls assigned to a particular role.

5.3.7. Unlock Process



Menu Path Security > Unlock Process

Functionality Processes are locked to prevent users from adding new data while a process is running. The Unlock Process allows you to view who locked the process and unlock it if necessary.

5.4. Data Tables – IMS Security

The figure below indicates the tables used by the Security module.

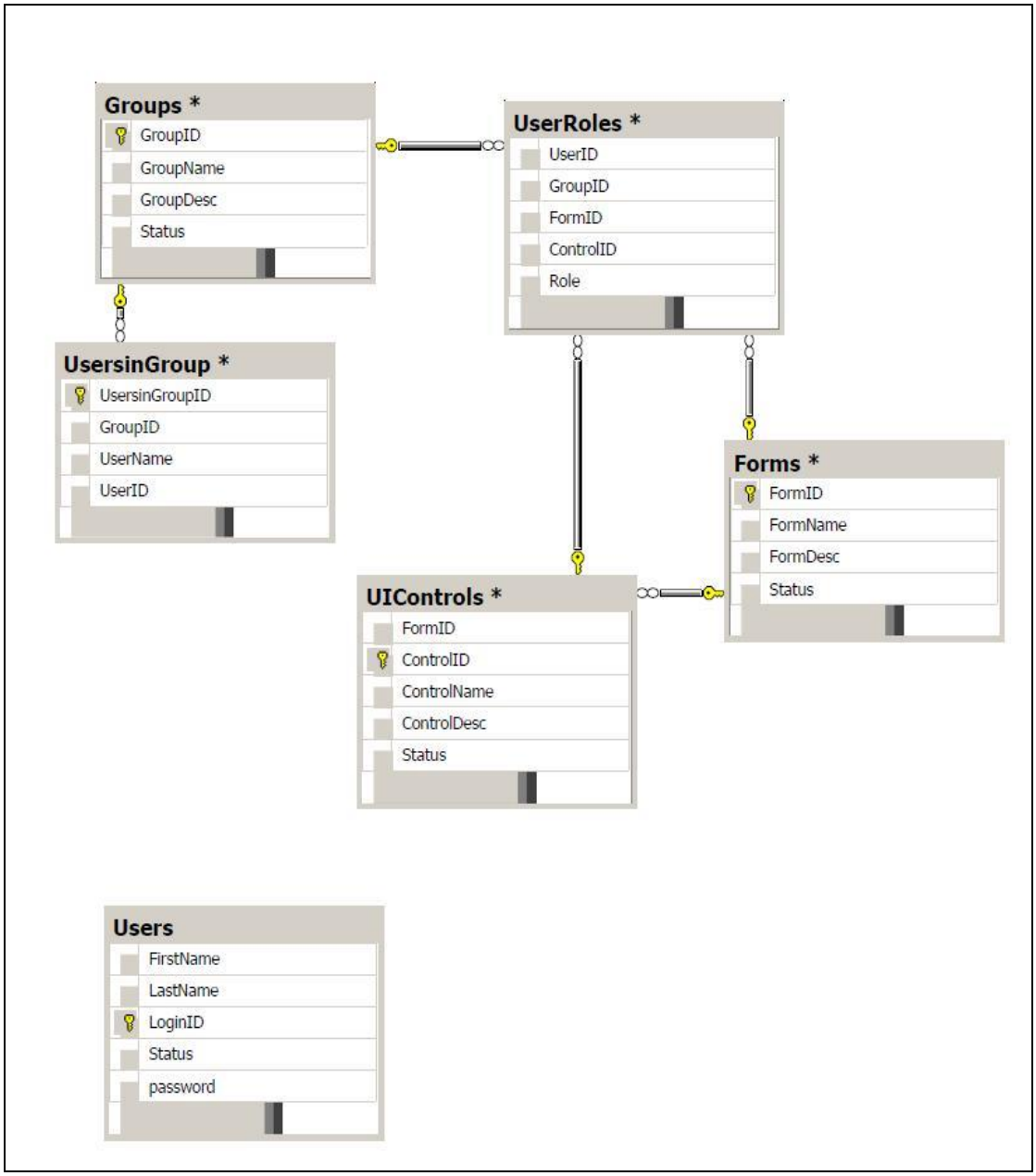


Figure 5 - 1 "Billing Management Data Tables"

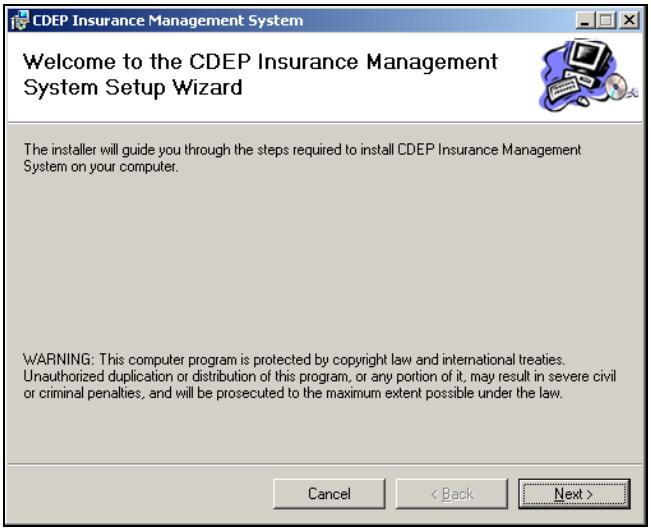
6. Client Installation Set Up and Deployment Procedures

The system administrator installs the Insurance Management Application once via a setup wizard, and then the application updates when the user logs on.

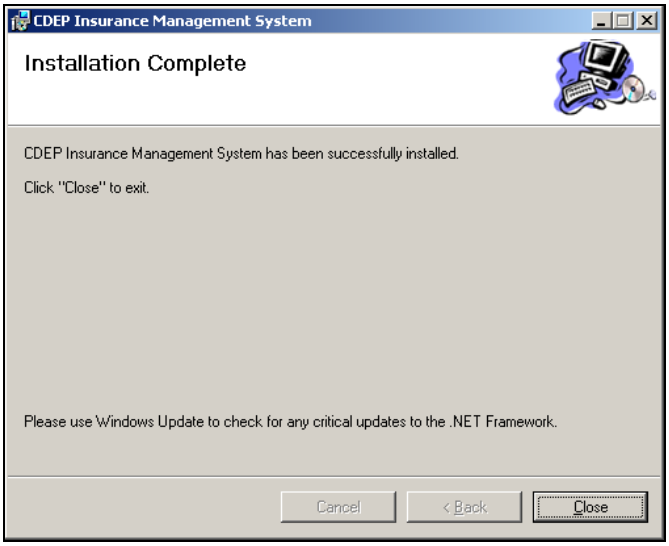
When the application is modified, the system updates automatically every time users log on.

To install the application:

1. Click the **setup.exe** file from the server.
   A wizard is displayed.



2. Follow the instructions on the Wizard until the application is installed.
   An Installation Complete screen is displayed.



3. Close the Wizard.

When the user logs onto the IMS application, the follow screens indicate the application is updating automatically.