Knowledge Transfer

DBS Financial Systems

Accounting 101 for R&D

Version: 0.4

Status: Draft

Updated: 10/11/2005

TABLE OF CONTENTS

REVISION HISTORY

Date	Version	Editor	Comments
9/9/05	0.1	Dan McFarland / Linda Muterspaugh	Based on 8/18/05 knowledge-transfer session by Dan McFarland.
9/19/05	0.2	Linda Muterspaugh/ Bill Froelich	Incorporates BSA comments and suggestions
9/26/05	0.3	Linda Muterspaugh/ Jen Buhman	Incorporates peer comments and suggestions
10/11/05	0.4	Linda Muterspaugh/ Dan McFarland/ Mildred Alvarez	Incorporates feedback from review meeting

1 INTRODUCTION

The goals of this session are to help R&D understand:

- how Operations and Accounting differ
- how the OPERATIONS_ACCOUNTING table in CSMS 3.0 (used by Operations) feeds data to the general ledger (GL) maintained by Accounting
- the difference between deferred revenue and recognized revenue
- how transactions are structured
- next steps for keeping pace with DBS growth

1.1 Scope

The scope of this document is limited to explaining, in simplified terms, DBS accounting basics. R&D needs to understand this information to be more effective in:

- processing system defects and enhancements that impact accounting data
- evaluating full-fledged accounting systems provided by third-party vendors

This document's scope does *not* include the technical information that R&D will need to work with current or future systems that impact accounting data.

1.2 References

DBS Accounting System Documentation: This series of diagrams analyzes the structure of the transactions that Accounting uses to account for DBS monies. The diagrams are currently being revised but an earlier version is located on the network at:

W:\Development\!Document_Library\DBS Systems Information\Accounting\Accounting Documentation

1.3 Terms and Abbreviations

Term or Abbreviation	Definition
breakage	Revenue that results when DBS no longer has an obligation to an end user and reclaims the value of the unused product. Typical conditions include the expiration of a product (no more minutes or grace days) or a disconnection. Also referred to as expired product or reclaimed product.
CDR	*Call Detail Record.* The carrier's record of the call's destination and duration. Used to debit the customer's contract and credit DBS with the revenue.
credit	Process that adds money to an account.
debit	Process that subtracts money from an account.
deferred revenue	Revenue that DBS receives in return for the promise to provide a service, such as providing airtime to an end user.
end user	The actual user of the contract minutes.
general ledger (GL)	A record of transactions organized by account. Typical subledgers include Cash, Accounts Receivables (money owed to the company), Accounts Payable (money the company owes to others), and Capital Transactions (such as loans or the proceeds from sales of assets).
GL transaction	The movement of funds from one account to another at a given point in time, using balanced debits and credits.
journal	In a database management system, the record of all stored data items that have been changed as a result of processing and manipulating the data. In Accounting systems, a journal number is used to identify the debit and credit threads used to represent a single transaction.
period	Unit of time used to group accounting data. Typical examples include month, quarter, six months, or year to date (YTD).
recognized revenue	Revenue that DBS can claim once it has fulfilled a promise to provide a service. There are two forms of recognized revenue: Dealer Revenue and Product Revenue.
Sarbanes/Oxley	Federal law designed to detect accounting irregularities. See http://www.sarbanes-oxley.com/

2 HOW OPERATIONS AND ACCOUNTING DIFFER

Operations is all about making the business work—selling cards to retailers, tracking call detail records, staffing customer service, and so on. Accounting, on the other hand, is charged with tracking the movement of money in a way that meets clear-cut, unchangeable legal rules.

Operations	Accounting
Represents the DBS interface to its customers.	Represents the DBS interface to tax and other legal agencies.
Represents a wide range of specific tasks, such as sending a fax to acknowledge a dealer's deposit.	Represents a limited set of pre-defined transactions.
Doesn't need to be "perfect" or inflexible in the way it implements DBS policies and procedures.	Needs to be perfect, provable, and predictable in the way it follows pre-defined rules for transactions. There is *no* room for creativity or variation.
Needs to track every single interaction that occurs between DBS and its customers.	Only needs to track transactions that have financial implications.

To emphasize the message in the last row—that Accounting only needs to track events in Operations that have a financial implication—see the figure below.

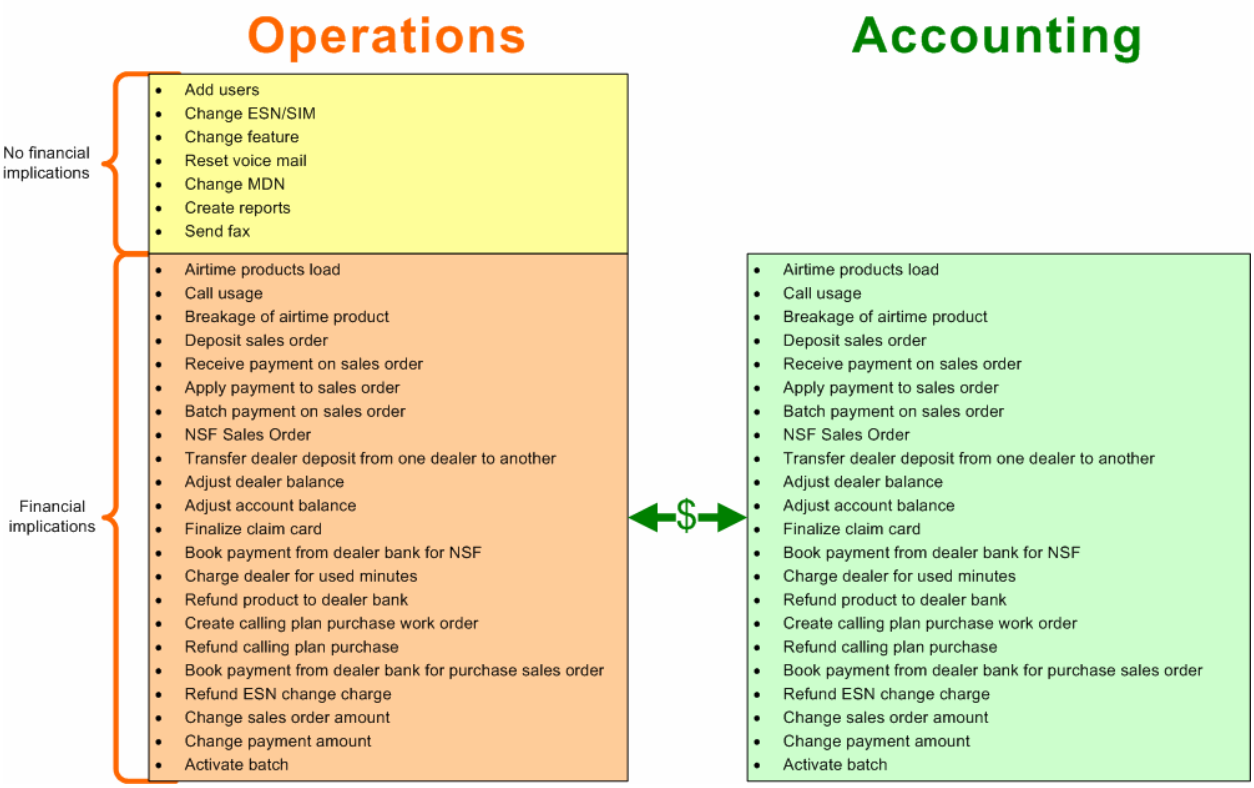

Figure 1 Operations events that need to be tracked by Accounting

To many, Accounting seems mysterious and intimidating. In simplified terms, though, we can think of Accounting as just tracing the movement of money between buckets. Let's see how this works if we use a ball to represent money.

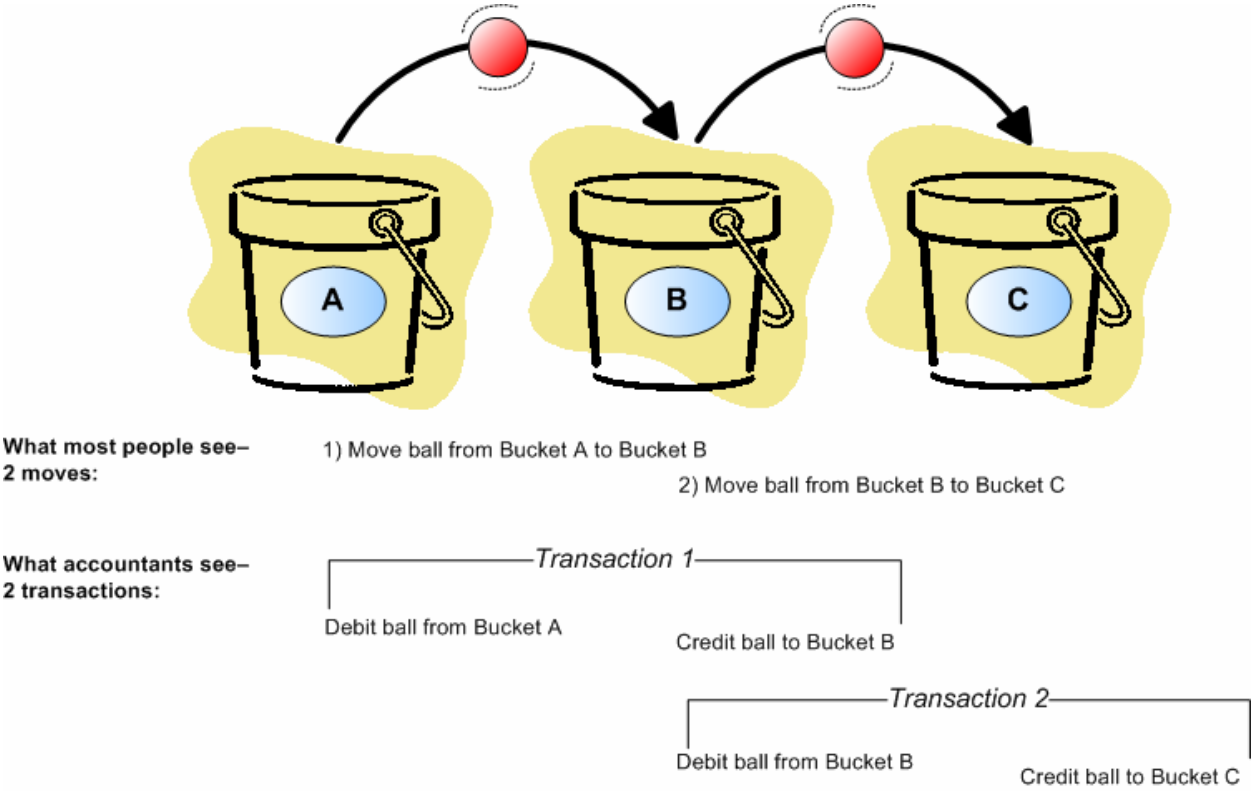

Figure 2 How accountants move money

3 HOW CSMS CREATES A POOL OF ACCOUNTING DATA

We've already seen that only certain operations have financial implications that need to be tracked by Accounting. How do we do track these operations for Accounting? Right now, one table in the CSMS database captures data about *transactions* that need to be tracked by Accounting: The OPERATIONS_ACCOUNTING table.

Note: The data stored in the OPERATIONS_ACCOUNTING table isn't an accounting system. It is a temporary pool of data structures—a holding tank—that can feed a real full-featured accounting package from a third-party vendor. DBS doesn't have the time, money, or expertise to create an accounting package with even the sophistication of Peachtree, which we've now outgrown.

3.1 Understanding Transactions

A *transaction* is a set of individual operations that are treated as a whole and exist under a set of predefined rules. The rules for a transaction can be divided into two categories:

- **General rules** that apply to *any* transaction
- **Specific rules** that apply to a specific *type of transaction*, such as making a deposit to a dealer bank.

3.1.1 General Rules for Transactions

As shown in Figure 2 on page 6, a transaction must consist of at least two operations—a debit and a credit. In addition, the debits and credits must happen at the exact same time and the sum of all the debits and credits must be zero. The operations that make up a transaction can be referenced individually or as a group. In Section 3.2, we will see examples of how these general rules have been implemented in the OPERATIONS_ACCOUNTING table.

Note: [Do we need a disclaimer that points the developer to a more complete set of requirements?]

3.1.2 Specific Rules for Transactions

R&D and Accounting have worked together to identify specific rules for the twenty-plus transactions that Accounting needs to track. These transactions are listed in Figure 1 on page 5. The rules for each of these transactions are detailed in the Accounting documentation listed in Section 1.2 on page 3 and an example is shown in Section 7 on page 17.

3.2 Analyzing the OPERATIONS_ACCOUNTING Table

In most database tables, you can insert, update, and delete records. This is *not* true of the OPERATIONS_ACCOUNTING table. *None* of the records stored in the OPERATIONS_ACCOUNTING table can ever be deleted or changed. Records are only created. This constraint makes sense if you think about the general rules for transactions, specifically the rule that the sum of all debits and credits need to sum to zero. If we wrote a program to update just a debit, for example, the sum would no longer be zero and the "books" wouldn't balance—an Accounting no-no.

The general rules for transactions are also supported by the way the OPERATIONS_ACCOUNTING table is divided into four segments, each of which has a specific purpose.

- **Transaction Tracking and Grouping Segment** ties all the records that represent an Accounting transaction back to the corresponding event in Operations.

- **Accounting Segment** contains the values that allow Accounting to categorize and summarize the data into "buckets."
- **Financial Values Segment** contains the debits and credits that must sum to zero.
- **Operationally Referenced Data Segment** contains fields that allow R&D to trace a record back to the code that created it and is used for troubleshooting.

3.2.1 Transaction Tracking and Grouping Segment

The purpose of the Transaction Tracking and Grouping Segment is to:

- tie an Accounting transaction back to its corresponding event in Operations.
- identify all the individual transaction records that need to be treated as a group.

3.2.1.1 The Fields

The fields that make up the Transaction and Grouping Segment include:

- PERIOD_ID. The PERIOD_ID field is a foreign key that links the record to the PERIOD table. A *period* is a duration of time that is defined for the purpose of archiving and summarizing data. (See Figure 3.) A records is assigned to a periods based on the value shown in INTERFACE_DATE and ACCT_TRANS_DATE.

  Note: The PERIOD_ID field is being implemented as part of a project to purge and archive data from CSMS; the project is scheduled to be completed by year end of 2005.

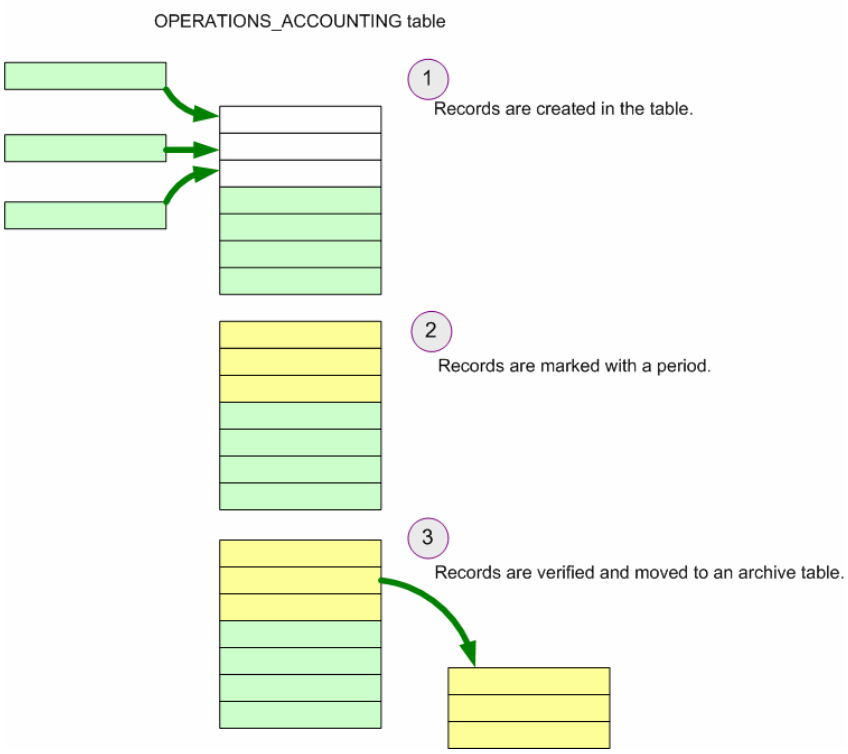

Figure 3 Using Periods to Group and Archive Records

- OPERATIONS_TRANSACTION_ID. This is a unique identifier for the record, which is assigned by the system.
- JOURNAL_HEADER_NUMBER. This is an identifier that is used to bind all records that have been created as the result of processing a transaction, which must consist of at least two or more records—a debit and a credit.
- INTERFACE_DATE. This is the effective date and time of the transaction shown on the CSMS interface.
- ACCT_TRANS_DATE. This is the date and time the record was created in the database, shown as Central Standard Time (CST).

3.2.1.2 The Rules

These are the primary rules that must be followed for the segment.

1. The INTERFACE_DATE and ACCT_TRANS_DATE fields must contain the same date and time. This rule allows us to be confident that there was no time during which funds could have been misdirected.
2. All records that have the same value in the JOURNAL HEADER NUMBER field must have the same value in the ACCT_TRANS_DATE field. If not, the records will be moved to an error table. This rules helps us be confident that we are consistent in the way we process all records that shared the same journal header number.
3. The processing for *all* records that have the same value in the JOURNAL HEADER NUMBER field must either succeed or fail. If processing for all the records succeeds, all of the records can be committed to the database. If processing for one of the records fails, none of the records can be committed to the database. Instead, all records must be rolled back from the database. This rule, again, helps us be confident that we are being consistent in processing all records that represent a transaction.

3.2.2 The Accounting Segment

The purpose of the Accounting Segment is to assign values to a record that can be used to categorize and summarize the raw data into “buckets” that are meaningful to the business.

The five fields are:

- ACCOUNT_CODE. The 7 alphanumeric digits of the account code typically describe a product.
- MARKET_CODE. The 3 digits of the market code identify the DBS-defined market.
- DEPARTMENT_CODE. The department code contains 3 digits.
- DEALER_CODE contains 5 digits.
- COMPANY_CODE contains 2 digits.

The five segments give DBS great flexibility in describing our products. For example, we could assign a unique ACCOUNT_CODE (product ID) to each of 60 different $35 cards or we could use a single ACCOUNT_CODE to group the cards, perhaps by color. Or we might have different types of products that affect how transactions are encoded. Or we could create dealer/account code pairs.

The Accounting Segment can also be used to summarize data in various ways. For example, we could group the products sold in a particular market and create a report that lets Sales, Marketing, Operations, Accounting, and the Executive Team see how the company is doing.

As an aside, five segments is the maximum number of segments that can be managed by most corporations. Only government agencies (like the IRS) need or can handle the complexity introduced by six accounting segments.

3.2.3 Financial Values Segment

3.2.3.1 The Fields

There are two numeric fields in the Financial Values Segment: DEBIT_AMOUNT and CREDIT_AMOUNT. All values are shown in U.S. dollars (USD). There are accounting conventions that determine whether debits and credits are shown as either positive or negative values.

3.2.3.2 The Rules

The important point to remember is that the sum of a debit and a credit is always zero.

3.2.4 Operationally Referenced Data Segment

The Operationally Referenced Data Segment contains two pairs of fields that allow R&D to trace a record back to the exact code that created it. These fields are not meaningful to Accounting, but, together, they allow R&D to troubleshoot errors that occur during certain time periods.

3.2.4.1 The Fields

- OPER_INTFACE_VERSION and OPER_INTFACE_ENTRY_POINT. If Accounting discovers an error in processing, R&D needs a way to identify the code that caused the problem. These two fields—OPER_INTFACE_VERSION and OPER_INTFACE_ENTRY_POINT—help R&D troubleshoot these errors. The OPER_INTFACE_VERSION field is used to represent the cvs tag of the code that created the record and the OPER_INTFACE_ENTRY_POINT field contains a number that is assigned to the line of code containing the insert statement that created the record.

  Note: When the OPERATIONS_ACCOUNTING table was designed, DBS only needed 5 characters to store the version number. Since then, R&D has implemented a naming convention that requires more than 5 characters. This design issue needs to be addressed.

- OPER_INTFACE_SUBSYSTEM and OPER_INTFACE_KEY. These two fields represent named pairs that tie the Accounting event back to an event in Operations. OPER_INTFACE_SUBSYSTEM contains the code for the subsystem that is performing the action. For example, SO represents a sales order or deposit to a dealer bank; WPP stands for Wireless Product Purchase—a product load. OPER_INTFACE_KEY contains the key for the record that contains the details about the action.

3.2.4.2 The Rules

There is a one-to-one correlation between the OPER_INTFACE_ENTRY_POINT and a unique location in the code. Developers tend to overlook this rule when they copy and paste code from one location to another and fail to realize that the value in the OPER_INTFACE_ENTRY_POINT needs to be changed to reflect the new location.

3.3 What the Table Doesn't Show

As an aside, you need to be aware that the table doesn't show all of the data generated by Operations. Some CSMS screens show data that *looks* like accounting data but is not. The Contract screen is a good example. This screen only shows Operations data—the airtime balance Operations needs to evaluate to determine whether the contract should be suspended. This information is *not* meaningful to Accounting, so it is not shown in the OPERATIONS_ACCOUNTING table.

4 HOW DBS MAKES ITS MONEY

4.1 A Simple Example

To understand the difference between deferred revenue and actual revenue (and how DBS makes its money), let's look at a simplified example that shows how a dealer's deposit is transformed to revenue.

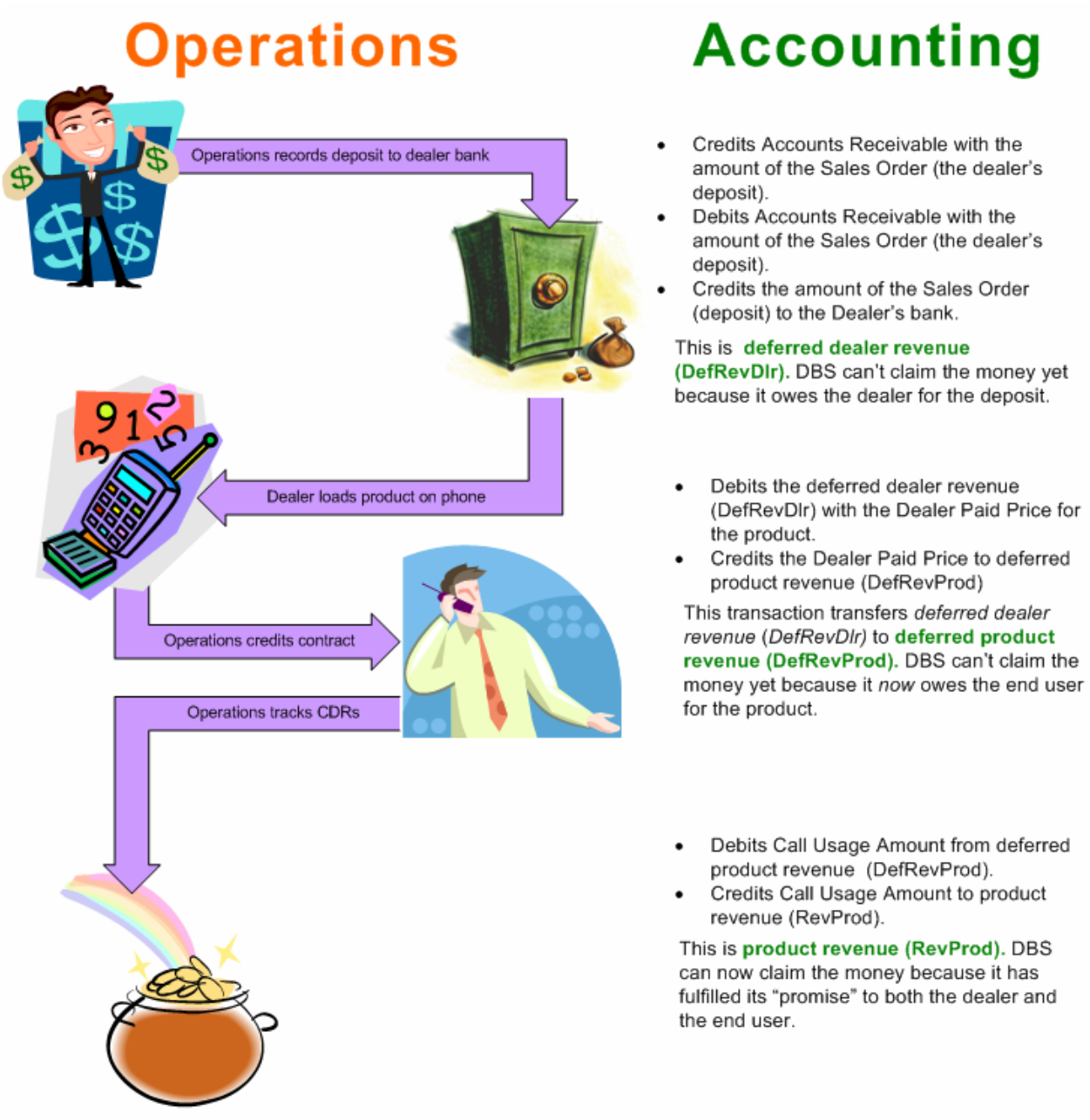

Figure 4 How DBS makes money

4.2 Types of Recognized Revenue

There are two basic types of recognized revenue—money that DBS gets to take to *its* bank:

- **Dealer Revenue.** This revenue is recognized when DBS makes an adjustment to the dealer bank (reclaims money) for some reason.
- **Product Revenue.** Product revenue is the most common form of recognized revenue and it occurs in two situations. Either end users use all the days and minutes on their contract to make calls (the example we saw in Section 4.1) or they *break* the contracts by letting the days expire without using all of the minutes they paid for. When this happens, DBS can reclaim the value of the unused minutes as recognized product revenue. This type of recognized product revenue is referred to as *breakage*, *reclaimed product*, or *expired product*—the terms mean the same thing.

5 PUTTING THE ACCOUNTABILITY IN ACCOUNTING

To be sure DBS is meeting its obligations, Accounting needs to track both deferred revenue and recognized revenue. How do you prove the accounting system is right and can account for 100 percent of all monies?

DBS can do this because it has analyzed all the accounting transactions to identify 15 to 20 predefined sequences of operations *(threads)* that have to happen the same way every time. For example, we can't process CDRs until the dealer loads a product on a phone, and we can't load a product until the dealer makes a deposit to the dealer bank. This lets us prove that we are 100 percent accountable for the money we take in (deferred revenue) and get to keep (recognized revenue).

The technical model for each of these transactions has been documented in *DBS Accounting System Documentation.* (See Section 1.2, References on page 3.) These models make acceptable assumptions; e.g., that Oracle will work as designed and so on. As you review these models, remember this mantra:

> All transaction threads have to be known, verifiable, and consistent every time they happen.

These transactions can be divided into two general categories:

- Transactions that track *deferred revenu*e, either from deferred dealer revenue and from deferred product revenue
- Transactions that track *recognized revenue*, either from call usage or from breakage

5.1 Accounting for Deferred Revenue

Generally speaking, the Accounting transactions that track *deferred revenue* are tightly coupled with Operations. There is a one-to-one relationship between a transaction in Operations and a transaction in Accounting. For example, when Operations uses CSMS to load a product, the system creates an equivalent Accounting transaction that happens in the same time. If the transaction succeeds, both the Operations and Accounting changes are committed to the database at the same time. If either fails, both the Operations and Accounting changes are rolled back.

5.2 Accounting for Recognized Revenue

The Accounting transactions that track recognized revenue—call usage and breakage—are more loosely coupled with transactions in Operations. There is no one-to-one relationship between Operations' receipt of a CDR, for example, and an Accounting transaction for that specific CDR. Instead, recognized revenue is calculated as the *percentage* of product sold vs. product used.

To understand the rationale for this approach, let's look at the recognized revenue that comes from call usage alone. DBS end users make between 4 and 5 million calls every day. Now try to estimate the processing power that would be needed to calculate product revenue on a call-by-call basis (the requirement for using a tightly coupled model). To further complicate matters, there are three ways to calculate the percentage of product used:

- as a percentage of anytime minutes sold
- as a percentage of access days sold
- as a percentage of the sum of anytime and off-peak minutes sold

And, to complicate matters even more, these percentages change constantly throughout the day, as customers purchase airtime and continue to make calls.

For all these reasons, Accounting calculates recognized product revenue as the percentage of all of the product that has been *used* (RevProd) vs. the value of all of the product that has been *sold* (DefRevProd). To meet the needs of Accounting and optimize processing requirements, a job runs twice a day to calculate recognized product revenue.

5.3 Accounting for Predictability: A Simple Example

How do we know that the *threads* of a transaction are known, verifiable, and consistent? We know this because all threads are documented in discrete technical models that have predefined start and end points.

To understand this, think of a thread as being a translucent pipe. If you put a ball in one end, you can't get two balls out the other end or have the ball come out of another pipe. The ball is either in the pipe or it is not and if it is in the pipe you should be able to tell exactly where it is.

To see how this works, let's take a look at the Sales Order "pipe" that is used to process a deposit to a dealer's bank. (See Figure 5.)

There are three key principles at work here:

- The transaction "inherits" values that are passed from Operations and uses the passed values to look up other values.
- We can perform calculations that create other transactions. In this example, some monies will go to the dealer's *unpaid deposit,* created when a dealer faxes a check to DBS and then follows up by mailing the actual check to DBS.
- All of the money that goes into the pipe is accounted for.

5.4 Accounting for Corrections

Accounting cannot change the logical thread of a string of transactions. And it can't update or delete a record once it is created. So how does Accounting correct a mistake, such as the clerk's typo that *credits* Dealer 10123 for a deposit that was received from Dealer 100321? It creates another set of predefined transactions to *debit* the deposit from Dealer 10123 and *credit* the deposit to Dealer 10321.

By the same token, we can use transactions to track intentional changes, such as changing a dealer's ID from 10321 to 10456. In this situation, we would use another of the predefined transactions to debit the entire balance in the dealer bank for Dealer 10321 and credit it to the dealer bank for Dealer 10456.

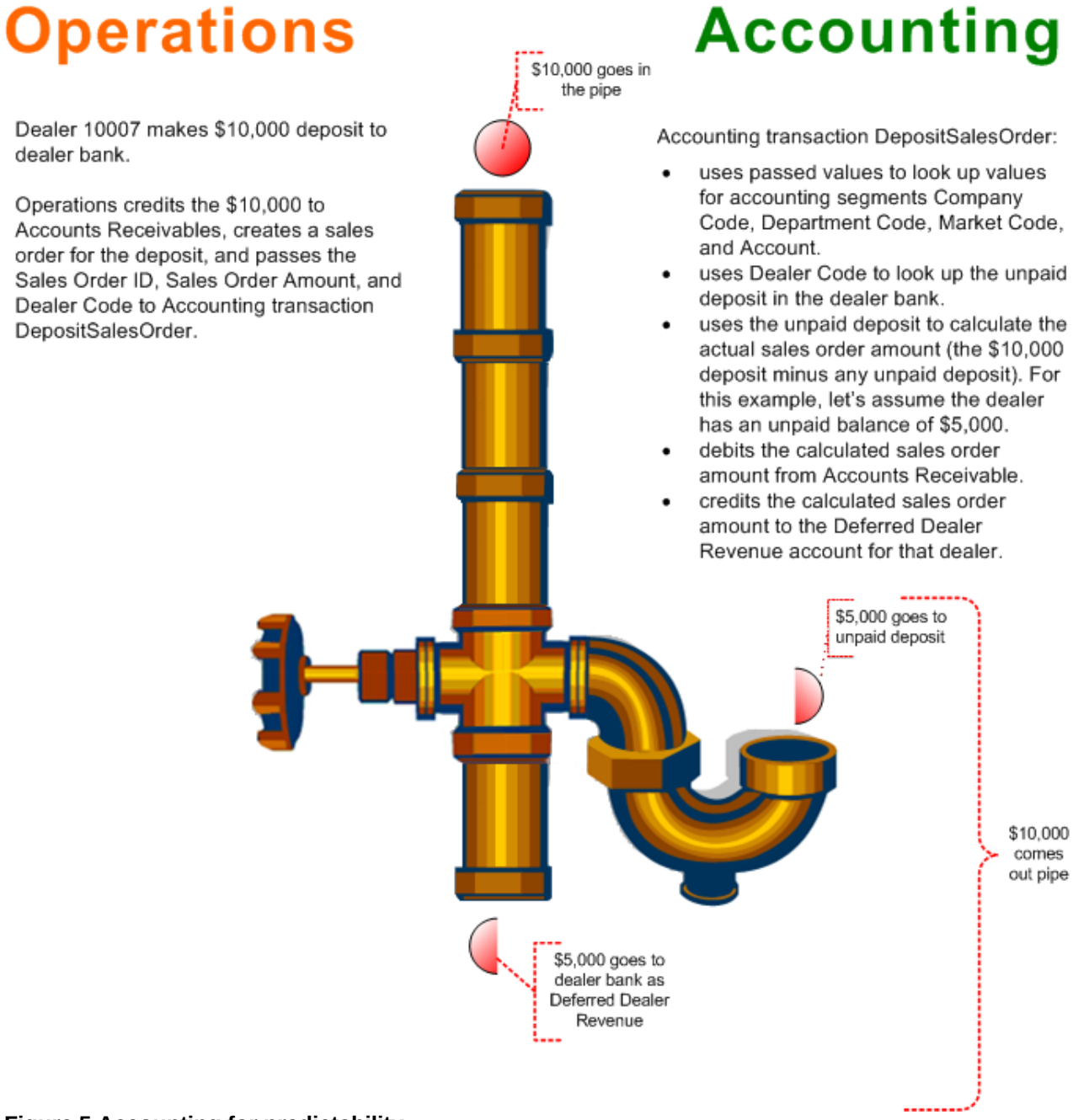

Figure 5 Accounting for predictability

6 WHERE DO WE GO FROM HERE?

DBS faces a few big issues in upgrading its accounting system.

1. Today, CSMS Operations uses a transaction that is *not* defined in the accounting system: the dealer's activation of a batch of cards. As a result, Operations can calculate the balance in the Dealer's Bank more accurately than Accounting. This is extremely bad and would definitely upset an auditor.
2. DBS has never closed a period in CSMS. (We didn't like the period-closing routine of our present accounting system, so we tossed it out.)
3. DBS needs to do a deep scrubbing of the CSMS data before we close a period in CSMS. This will help us be sure we comply with legal requirements, such as *Sarbanes/Oxley*, at the time of closing and it will minimize the amount of data we need to maintain in the live operations system.

The good news:

1. Ours is a well-designed system that can interface with any accounting system and be used to upload accounting data.
2. We can use the OPERATIONS_ACCOUNTING table from CSMS for a trial close-out of a period.

Five of the twelve items on the Comptroller's list have to do with the need for a better accounting system:

1. Add the missing transaction for activating a batch. This transaction is missing from *DBS Accounting System Documentation* and is not captured in the OPERATIONS_ACCOUNTING table.
2. Create the ability to close a period.
3. Create different sets of reports or teach users to use existing reports to prepare for closing a period.
4. Implement a more robust, full-featured accounting system.
5. Simplify auditing by implementing a backward sniffing tool that is based on the accounting system.

7 APPENDIX: A CLOSER LOOK AT THE TRANSACTION DIAGRAMS

If you are a developer, you will need to become familiar with the transaction diagrams of the Accounting transactions. This appendix contains a sample. If you need access to the entire set of diagrams, contact your manager.

Beginning at the top left corner, we see an arrow that represents the SO (Sales Order) Account Packet that Operations passes to Accounting and the three values within the packet:

- SalesOrderID is used to look up the accounting segments of CompanyCode, DeptCode, MarketCode, and ARAccount (Accounts Receivable).
- SalesOrderAmount is used to determine how much to debit from Accounts Receivable. This amount is also used in calculated the amount of money that will be credited to the dealer's bank.
- DealerCode is used to determine whether there is an unpaid balance that needs to be subtracted from the sales order (deposit to the dealer's bank).

Continuing to the right, we see two rules. The first rule indicates the field we can use to trace back to the Operations transaction. The second rule shows that the amount of money that we credit to the dealer's bank is a calculated value, determined by subtracting any unpaid deposit from the amount of the sales order (deposit).

Below these rules, the diagram shows two tables that represent the threads of this transaction.

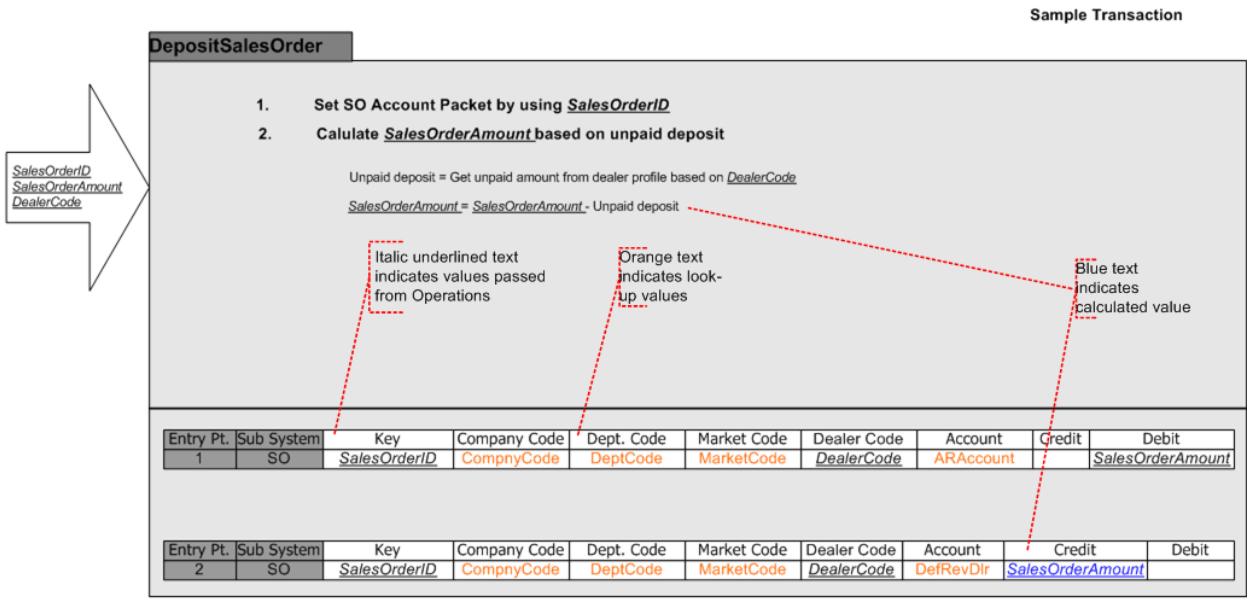

Figure 6 A sample accounting transaction diagram