Oracle® Financial Services

Reference Guide

Release 12

Part No. B31078-01

December 2006

ORACLE®

Oracle Financial Services Reference Guide, Release 12

Part No. B31078-01

Primary Author: Vijay Tiwary

Contributing Author: Mayur Polepalli

Contributor: Amit Budhiraja, Lokesh Garg, Jack Hickox, Angel Lafont, Essan Ni, Christopher Spofford

Contents

3 The Ledger Migration Process

4 Rate Conversion

5 Cash Flow Edit Logic

6 Cash Flow Calculations

7 Cash Flow Dictionary

Index

Send Us Your Comments

Oracle Financial Services Reference Guide, Release 12

Part No. B31078-01

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on Oracle MetaLink and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

Intended Audience

Welcome to Release 12 of the *Oracle Financial Services Reference Guide.*

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Computer desktop application usage and terminology

If you have never used Oracle Applications, we suggest you attend one or more of the Oracle Applications training classes available through Oracle University.

See Related Information Sources on page xiii for more Oracle Applications product information.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site

at http://www.oracle.com/accessibility/ .

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

1 Transfer Pricing Concept

This chapter describes the transfer pricing concept and the traditional and matched rate approaches to transfer pricing. A description of how matched rate transfer pricing overcomes the shortcomings of the traditional approaches is also provided. The chapter ends with a description of the role of matched rate transfer pricing in the evaluation of interest rate risk.

2 Transfer Pricing Option Costs

This chapter describes how Oracle Transfer Pricing goes about calculating option costs. The chapter begins with an introduction to option costs and subsequently describes option cost theory and the calculation architecture.

3 The Ledger Migration Process

This chapter discusses the process for generating credits or charges, for funds provided or used, and their migration to the Management Ledger table. The chapter provides a detailed description of how the information required for generating these credits or charges originates through transfer rates and option cost processing from the Customer Account tables and how the results are inserted into the Management Ledger table.

4 Rate Conversion

This chapter describes how Oracle Transfer Pricing translates interest rates from their initial formats into formats that can be used by the application.

5 Cash Flow Edit Logic

This chapter describes the set of logical validations that are performed, and the order in which they should be performed, on the Account table data during a Cash Flow Edits rule run.

6 Cash Flow Calculations

This chapter discusses cash flow calculation concepts and processes.

7 Cash Flow Dictionary

This chapter lists and describes the account table columns that store cash flow data and are accessed by Oracle Transfer Pricing during cash flow processing and edits.

The chapter also provides detailed information on how to correctly populate account table columns with instrument data, including column descriptions, usage, and recommended default values for the cash flow processing columns.

Related Information Sources

This document is included on the Oracle Applications Document Library, which is supplied in the Release 12 DVD Pack. You can download soft-copy documentation as PDF files from the Oracle Technology Network at http://otn.oracle.com/documentation, or you can purchase hard-copy documentation from the Oracle Store at http://oraclestore.oracle.com. The Oracle E-Business Suite Documentation Library Release 12 contains the latest information, including any documents that have changed significantly between releases. If substantial changes to this book are necessary, a revised version will be made available on the online documentation CD on Oracle *MetaLink*.

If this guide refers you to other Oracle Applications documentation, use only the Release 12 versions of those guides.

For a full list of documentation resources for Oracle Applications Release 12, see Oracle Applications Documentation Resources, Release 12, Oracle*MetaLink* Document 394692.1.

Online Documentation

All Oracle Applications documentation is available online (HTML or PDF).

- **PDF** - PDF documentation is available for download from the Oracle Technology Network at http://otn.oracle.com/documentation.

- **Online Help** - Online help patches (HTML) are available on Oracle*MetaLink*.

- **Oracle MetaLink Knowledge Browser** - The Oracle*MetaLink* Knowledge Browser lets you browse the knowledge base, from a single product page, to find all documents for that product area. Use the Knowledge Browser to search for release-specific information, such as FAQs, recent patches, alerts, white papers, troubleshooting tips, and other archived documents.

- **Oracle eBusiness Suite Electronic Technical Reference Manuals -** Each Electronic Technical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing applications and integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on OracleMetaLink.

Related Guides

You should have the following related books on hand. Depending on the requirements of your particular installation, you may also need additional manuals or guides.

Oracle Applications Installation Guide: Using Rapid Install:

This guide provides information about using the Rapid Install utility to install Oracle Applications Release 12, or as a part of an upgrade from Release 11i to Release 12. Discusses Standard and Express installations, fresh or Vision Demo database installations, as well as techstack and product upgrades.

Oracle Applications Maintenance Procedures:

This guide describes how to use AD maintenance utilities to complete tasks such as compiling invalid objects, managing parallel processing jobs, and maintaining snapshot information. Part of Maintaining Oracle Applications, a 3-book set that also includes Oracle Applications Patching Procedures and Oracle Applications Maintenance Utilities.

Oracle Applications Maintenance Utilities:

This guide describes how to run utilities, such as AD Administration and AD Controller, used to maintain the Oracle Applications file system and database. Outlines the actions performed by these utilities, such as monitoring parallel processes, generating Applications files, and maintaining Applications database entities. Part of Maintaining Oracle Applications, a 3-book set that also includes Oracle Applications Patching Procedures and Oracle Applications Maintenance Procedures.

Oracle Applications Patching Procedures:

This guide describes how to patch the Oracle Applications file system and database using AutoPatch, and how to use other patching-related tools like AD Merge Patch, OAM Patch Wizard, and OAM Registered Flagged Files. Describes patch types and structure, and outlines some of the most commonly used patching procedures. Part of Maintaining Oracle Applications, a 3-book set that also includes Oracle Applications Maintenance Utilities and Oracle Applications Maintenance Procedures.

Oracle Applications Upgrade Guide: Release 11i to Release 12:

This guide provides information for DBAs and Applications Specialists who are responsible for upgrading a Release 11i Oracle Applications system (techstack and products) to Release 12. In addition to information about applying the upgrade driver, it outlines pre-upgrade steps and post-upgrade steps, and provides descriptions of product-specific functional changes and suggestions for verifying the upgrade and reducing downtime.

Oracle Alert User's Guide:

This guide explains how to define periodic and event alerts to monitor the status of your Oracle Applications data.

Oracle Application Framework Developer's Guide:

This guide contains the coding standards followed by the Oracle Applications

development staff to produce applications built with Oracle Application Framework. This guide is available in PDF format on Oracle*MetaLink* and as online documentation in JDeveloper 10*g* with Oracle Application Extension.

Oracle Application Framework Personalization Guide:

This guide covers the design-time and run-time aspects of personalizing applications built with Oracle Application Framework.

Oracle Applications Concepts:

This book is intended for all those planning to deploy Oracle E-Business Suite Release 12, or contemplating significant changes to a configuration. After describing the Oracle Applications architecture and technology stack, it focuses on strategic topics, giving a broad outline of the actions needed to achieve a particular goal, plus the installation and configuration choices that may be available.

Oracle Applications Developer's Guide:

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards for Forms-Based Products*. It also provides information to help you build your custom Oracle Forms Developer forms so that they integrate with Oracle Applications.

Oracle Applications Flexfields Guide:

This guide provides flexfields planning, setup, and reference information for the Oracle Applications implementation team, as well as for users responsible for the ongoing maintenance of Oracle Applications product data. This guide also provides information on creating custom reports on flexfields data.

Oracle Applications System Administrator's Guide Documentation Set:

This documentation set provides planning and reference information for the Oracle Applications System Administrator. *Oracle Applications System Administrator's Guide - Configuration* contains information on system configuration steps, including defining concurrent programs and managers, enabling Oracle Applications Manager features, and setting up printers and online help. *Oracle Applications System Administrator's Guide - Maintenance* provides information for frequent tasks such as monitoring your system with Oracle Applications Manager, managing concurrent managers and reports, using diagnostic utilities, managing profile options, and using alerts. *Oracle Applications System Administrator's Guide - Security* describes User Management, data security, function security, auditing, and security configurations.

Oracle Applications User's Guide:

This guide explains how to navigate, enter data, query, and run reports using the user interface (UI) of Oracle Applications. This guide also includes information on setting user profiles, as well as running and reviewing concurrent requests.

Oracle Integration Repository User's Guide:

This guide covers the employment of Oracle Integration Repository in researching and deploying business interfaces to produce integrations between applications.

Oracle Web Applications Desktop Integrator Implementation and Administration Guide:

Oracle Web ADI brings Oracle E-Business Suite functionality to a spreadsheet where familiar data entry and modeling techniques can be used to complete Oracle E-Business Suite tasks. You can create formatted spreadsheets on your desktop that allow you to download, view, edit, and create Oracle E-Business Suite data that you can then upload. Use this guide to implement Oracle Web ADI and for information on defining mappings, layouts, style sheets, and other setup options.

Oracle Workflow Administrator's Guide:

This guide explains how to complete the setup steps necessary for any product that includes workflow-enabled processes. It also describes how to manage workflow processes and business events using Oracle Applications Manager, how to monitor the progress of runtime workflow processes, and how to administer notifications sent to workflow users.

Oracle Workflow API Reference:

This guide describes the APIs provided for developers and administrators to access Oracle Workflow.

Oracle Workflow Developer's Guide:

This guide explains how to define new workflow business processes and customize existing Oracle Applications-embedded workflow processes. It also describes how to define and customize business events and event subscriptions.

Oracle Workflow User's Guide:

This guide describes how users can view and respond to workflow notifications and monitor the progress of their workflow processes.

Oracle Business Intelligence Discoverer Administration Guide:

Use this guide to find out how to set up and maintain a Discoverer system after installation. It covers how to use Discoverer Administrator to: create and maintain End User Layers; to set up business areas, folders and items; to help users find information by defining joins, calculated items, and conditions; and to improve Discoverer performance.

Oracle Business Intelligence Discoverer Plus User's Guide:

Use this guide to find out how to retrieve and analyze data by creating worksheets and charts, and how to publish those results. It covers the most common tasks you will perform with Discoverer Plus (for example, drilling and pivoting), along with reference information and useful examples. It includes an appendix containing detailed calculation examples.

Oracle Business Intelligence Discoverer Viewer User's Guide:

Use this guide to find out how to analyze data in worksheets that have already been created in Discoverer Plus. It covers the most common tasks you will perform with Discoverer Viewer (for example, drilling and pivoting), along with reference information and useful examples.

Oracle Embedded Data Warehouse Implementation Guide:

This guide describes how to implement Embedded Data Warehouse, including how to set up the intelligence areas.

Oracle Embedded Data Warehouse Install Guide:

This guide describes how to install Embedded Data Warehouse, including how to create database links and create the end user layer (EUL).

Oracle Embedded Data Warehouse User Guide:

This guide describes how to use Embedded Data Warehouse reports and workbooks to analyze performance.

Oracle Enterprise Performance Foundation User's Guide:

This guide describes Oracle Enterprise Performance Foundation, an open and shared repository of data and business rules that provides the framework for all of the applications in the Corporate Performance Management set of products. It describes the product features that allow you to manage repository metadata and enable you to generate management reports and perform analyses.

Oracle Financial Services Implementation Guide:

This guide provides information about setting up Oracle Financial Services (OFS) applications in Release 12.

Oracle Financial Services Reporting Administration Guide:

This guide describes the reporting architecture of Oracle Financial Services applications in Release 12, and provides information on how to view these reports.

Oracle General Ledger Implementation Guide:

This guide provides information on how to implement Oracle General Ledger. Use this guide to understand the implementation steps required for application use, including how to set up Accounting Flexfields, Accounts, and Calendars.

Oracle General Ledger Reference Guide:

This guide provides detailed information about setting up General Ledger Profile Options and Applications Desktop Integrator (ADI) Profile Options.

Oracle General Ledger User's Guide:

This guide provides information on how to use Oracle General Ledger. Use this guide to learn how to create and maintain ledgers, ledger currencies, budgets, and journal entries. This guide also includes information about running financial reports.

Oracle Profitability Manager User's Guide:

This guide describes Profitability Manager, which provides a rich set of features that support complex models to analyze your business. These features include a powerful allocation engine that supports many allocation methodologies, Activity-Based Management calculations that provide activity costs, rolled up costs and statistics, activity rates, and cost object unit costs, and customer profitability calculations to consolidate customer accounts, aggregate customer data, and determine profitability results.

Oracle Transfer Pricing User Guide:

This guide contains the information you need to understand and use Oracle Transfer Pricing, including how to generate transfer rates and option costs for your product portfolio and determine account level match-funded spreads.

Integration Repository

The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite's business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Applications tables are interrelated, any change you make using an Oracle Applications form can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a

record of changes.

1

Transfer Pricing Concept

This chapter describes the transfer pricing concept and the traditional and matched rate approaches to transfer pricing. A description of how matched rate transfer pricing overcomes the shortcomings of the traditional approaches is also provided. The chapter ends with a description of the role of matched rate transfer pricing in the evaluation of interest rate risk.

This chapter covers the following topics:

- Overview of Transfer Pricing
- Traditional Approaches to Transfer Pricing
- A New Approach to Transfer Pricing
- How Matched Rate Transfer Pricing Works
- Evaluating Interest Rate Risk

Overview of Transfer Pricing

Over the past few decades, financial institutions, such as banks, have evolved as semiautonomous lines of business. Consequently, management requires separate income statements and balance sheets for each line of business to assess its performance. However, creating separate income statements and balance sheets requires the division of the net interest income among the business units. Oracle Transfer Pricing fulfills this need. Transfer pricing is a mechanism for dividing the net interest income of a financial institution (such as a bank) among its constituent business units (such as the deposit, treasury, and the credit groups).

Transfer pricing makes use of transfer rates to divide the net interest income into manageable components by separately identifying the spread earned from interest rate risk and the spread earned from risks managed by the lines of business such as credit risk. The transfer rate for funds is an interest rate representing the value of those funds to a financial institution, that is, the interest rate at which the financial institution can buy or sell those funds in open market.

The transfer rate provides a benchmark for determining whether the yield on a loan (an asset), is enough to cover the associated credit risk and operating cost, besides the cost of acquiring the funds. In addition, a transfer rate for funds allows you to compare the total cost of each source of funds, such as deposits (a liability), to other funding opportunities, for example money or capital market funds. In effect, you use a transfer rate to measure the profit contribution of an asset or liability.

The following table shows a typical bank balance sheet.

Typical Bank Balance Sheet

Asset	Liability
Less Transfer Cost of Funds/Spread on Assets	Less Cost of Funds/Spread on Liability
Less Operating Cost/Profit Contribution	Less Operating Cost/Profit Contribution

Most large banks have recognized the value of transfer pricing and it has been a part of their performance measurement systems for years. However, the gains from adopting a transfer pricing framework depend on the maturity of methodology being used.

Traditional Approaches to Transfer Pricing

Banks following the traditional approaches to transfer pricing applied a single transfer rate to the net volume of funds generated or consumed by a business unit. They generally used either the average or the marginal cost of funds as the single transfer rate.

Until the 1960s, banks generally used the average cost of funds as the single transfer rate, primarily for loan pricing. If the yield on a loan was higher than the average cost of funds, banks believed that the loan had a positive spread and made the loan.

Over time, the problem with this approach became obvious. Regulated low rate deposits, such as Demand Deposit Accounts (DDA) and Savings Accounts, held the average cost of funds for many banks at a level well below the cost of the new funds. As a result, spreads on new volumes were nowhere near what had been expected. Moreover, the low average cost of funds tempted many banks to underprice loans, sometimes to the point where the true spreads on new volumes were negative.

Consequently, even a stable rate environment was potentially dangerous for banks using the average cost approach to transfer pricing because the balance sheet could grow while earnings dropped.

Recognizing that the use of an average cost of funds could result in unprofitable growth, most banks concluded they should use a transfer price reflecting their real cost of incremental funds. Typically, these banks used the cost of 30 or 90 day certificates of deposit (CDs) as the cost of marginal funds.

Pitfalls of the Traditional Approaches

The shortcomings of the transfer pricing approaches that advocate the use of a single transfer rate are:

- **Potential for Inadvertent Unprofitable Growth:** Banks assumed that using the marginal cost of funds would make it almost impossible to add volume at a negative spread. This is a fair assumption but it only applies to times when interest rates are stable.

- **Rate Risk Trap:** The single, marginal funds transfer rate led some financial institutions into a rate risk trap. In the 1960s and 1970s, because the yield curve was normal, long term assets offered the largest spreads against a 30 or 90 day transfer rate. So some banks, and almost the entire savings and loan industry, borrowed short and lent long. Interest rates skyrocketed in 1979 and into the 1980s, and consequently the margins disappeared.

- **Loss of the Credibility of Performance Measurement Systems:** Most banks were able to avoid the extreme interest rate risk exposure, which nearly destroyed the savings and loan industry.

  However, the use of a single marginal transfer rate undermined the credibility of performance measurement systems. Most line of business managers found that their bottom lines fluctuated wildly with interest rates. Since market interest rates were obviously beyond the control of line managers, they increasingly viewed profit goals for their units with skepticism.

- **Loss of Managerial Value:** The traditional approaches failed to offer any generally accepted (or politically acceptable) method for determining the net interest contribution of the different business units of a bank. Consequently, business unit profitability reporting lost its managerial decision-support value.

In summary, the traditional approaches to transfer pricing were acceptable when interest rates were stable. However, they lost most of their decision-supporting value once rates became volatile.

Related Topics

Overview of Transfer Pricing, page 1-1

A New Approach to Transfer Pricing

As the shortcomings of the traditional transfer pricing systems became obvious, the financial services industry began to search for alternatives. The best solution was developed and implemented by a few leading financial institutions in 1979 and 1980. This approach, called matched rate transfer pricing, uses multiple transfer rates. Assets and liabilities are given transfer rates that reflect their specific maturity and repricing

characteristics.

Matched rate transfer pricing resolves the problems inherent in traditional methodologies by:

- Clearly showing whether new volumes have a positive spread by using a marginal rate. This eliminates the potential for inadvertent unprofitable growth.
- Identifying potential rate risk traps in advance using a marginal rate. In addition, the exposure of a bank to interest rate risk is identified and measured in a manner that makes it easier to manage.
- Ensuring that the performance measurement system is consistent, fair, and credible by using a transfer rate that reflects real funding opportunities currently available to the bank.

Matched rate transfer pricing achieves these objectives by dividing the interest rate spread into three components: credit spread, funding spread, and rate risk spread.

Example of Dividing Interest Rate Spread

Suppose a retail financial institution, a bank for example, relies on a retail customer base for low cost funds that have interest rates lower than funds purchased in money markets. It uses these funds to make loans that have a yield much higher than what the financial institution would pay for funds having the same maturity.

Consider a consumer loan, that yields 200 basis points higher than what the financial institution would pay for funds having the same maturity. Suppose the bank decides to fund the loan with matched maturity funds, say, certificates of deposit that costs 100 basis points less than similar maturity funds purchased in money markets. Then, the bank will have a total interest rate spread of 300 basis points.

Matched rate transfer pricing divides this interest rate spread as follows. While the loan yields 200 basis points more than matched funding costs (transfer rate), the funds cost 100 basis points less than other alternatives (transfer rate). Therefore, the total spread of 300 basis points is the sum of a funding spread (transfer rate - cost of funds) of 100 points and a credit spread (yield on loans - transfer rate) of 200 points.

However, if the financial institution funds the consumer loan with shorter term deposits, then the spread would be larger than 300 basis points. The added spread results from taking interest rate risk (borrowing short and lending long) and is called rate risk spread. The three components of the interest rate spread can be seen by plotting the loan and deposit against the yield curve. However, the portion of total spread derived from taking interest rate risk can be volatile.

Advantages of Matched Rate Transfer Pricing

The main advantages of matched rate transfer pricing are as follows:

- **Stabilization of Business Unit Margins:** The use of multiple matched transfer rates stabilizes the margins of the different business units. Since assets and liabilities are either funded or sold to transfer pools with corresponding maturities or repricing periods, swings in interest rates do not affect the spread. In addition, the division of the interest rate spread into credit, funding, and rate risk spreads ensures that the bottom line for a business unit reflects only that business and is within the control of the line management.

- **Decision Support:** Under the matched rate transfer pricing approach, the bottom line for a business unit is a fair basis for its performance measurement and management. For example, if some types of loans consistently fail to cover the cost of funds, operating costs, and the credit risk, there is no reason to make those loans. It would be more profitable to buy bonds, or to find other, more profitable lending opportunities. This is the reason why many banks no longer make small installment loans.

  Similarly, if the operating costs of gathering low cost consumer deposits are too high, it may be more economical to purchase funds in money markets. This explains the growing number of branch closures, as well as the imposition of increasingly higher minimum balances on some types of consumer deposits.

- **Identifying Exposure to Interest Rate Risk:** Using matched rate transfer pricing, banks can identify their exposure to interest rate risk and its impact on their current earnings. In addition, the banks can isolate the spread from rate risk exposure from their total spreads. This helps them clearly determine the profitability of their business units. Also, banks have found that the interest rate risk becomes increasingly manageable when isolated in a separate business unit. Under the matched rate transfer pricing approach, the rate risk exposure, and its impact on current earnings, is revealed in a new profit center called Treasury. See: How Matched Rate Transfer Pricing Works, page 1-5.

In summary, matched rate transfer pricing works well even when interest rates are volatile. It provides an approach to performance measurement that meets the decision making needs of both line managers (consistency, fairness, controllability) and executive managers (accuracy, flexibility). The financial services industry has recognized these benefits. Consequently, there is an increasing number of financial institutions that have either implemented, or are in the process of implementing performance measurement systems based on matched rate transfer pricing.

Related Topics

Overview of Transfer Pricing, page 1-1

How Matched Rate Transfer Pricing Works

Matched rate transfer pricing is often administered by the Treasury. The Treasury

conceptually buys the funds from the deposit gathering group and sells them to the credit group.

Line officers get a rate quote representing either the cost of the funds they want to lend, or the value of the deposits they are gathering. The spread between this quoted rate and the interest rate on the asset or liability is fixed at a known level and maintained for the life of the asset or liability. Any fluctuation in this spread, whether caused by changes in the asset or liability yield curves or in the funds transfer yield curve, gets accumulated at the Treasury level.

The Treasury can manage the fluctuation in the spread in several ways, for example:

- Maintain a discretionary portfolio of assets and liabilities with the sole purpose of offsetting the risk that has been transferred from other business units.
- Use off-balance sheet transactions, such as swaps and futures, to hedge risk.

Matched rate transfer pricing requires more accounting discipline than traditional transfer pricing approaches. However, it is a straightforward process and is applied in a logical manner, using standard principles of dual-entry accounting.

Matched Rate Transfer Pricing Example

Suppose a line officer wants to make a loan, and is trying to decide on its pricing. The line officer is given a cost of funds that reflects the maturity and repricing characteristics of the loan. If it is to be a long-term, fixed rate loan, the bank quotes the cost of the long-term funds that can be used to match that loan. Conversely, if the loan is to be short term, the line officer is quoted a short-term rate.

If the yield curve is normal, the transfer rate for a short term loan is less than the rate for a long-term loan. The line officer then figures out how to price the loan to attain a target spread over the quoted cost of funds.

When the loan is booked:

- The business unit of the line officer books a shadow liability equal in volume to the size of the loan, having a cost that equals the transfer rate that was quoted. This accounting transaction balances the books of the business unit, and locks in a spread as long as the loan stays on the books.
- The books of the corporation must be balanced. Banks do this by creating a shadow asset with equal size and rate to the shadow liability. This shadow asset is housed in a separate business unit, usually Treasury.

The same type of accounting is applied to liabilities also. This type of accounting divides the bank's profits into three components: lending profit, deposit gathering profit, and rate risk profit. These three components add up to the total profit of the bank.

To sum up, under the matched rate transfer pricing approach, banks attach a matched

transfer rate to an asset or liability when it is booked, using a standard, double-entry accounting approach. This transfer rate remains constant over the life of the asset or liability, stabilizing the spread for the line of business.

Related Topics

Overview of Transfer Pricing, page 1-1

Evaluating Interest Rate Risk

Matched rate transfer pricing divides the net interest income of your institution into three components: lending, deposit, and the rate risk profit (or loss). The rate risk profit is derived by subtracting all credits for funds (funding center expense) from all charge for funds (funding center income). See: How Matched Rate Transfer Pricing Works, page 1-5.

A net positive number implies that part of your interest margin is a result of any rate bets (or rate risk) your institution has taken. A negative number implies that you have incurred a loss due to rate risk.

Current and Embedded Rate Risks

The total rate risk profit (or loss) figure is made up from two sources:

- **Current Rate Risk Profit:** The result of rate risk inherent in your current exposure. You can actively manage this profit through effective Asset/Liability management.
- **Embedded Rate Risk Profit:** The result of interest rate bets. You can no longer manage this component of earnings because the relationships are contractual. All you can do is to wait it out.

Embedded Rate Risks Example

Suppose a bank, on day one, raises $1,000 in the form of a one-year certificate of deposit at 4%. If the wholesale (open market) alternative to one year funds costs 5% then, the matched transfer rate is 5%.

The bank then lends the $1,000 in the form of a five year nonamortizing (bullet) loan at 10%. If the cost of five-year wholesale funds is 8% then the matched transfer rate for five year funds is 5%.

This table shows the components of the bank's interest rate margin on day one:

Components of Interest Rate Margin for the Bank on Day One

Income Statement Component	Rate	Transfer Rate	Spread
Asset	10.00%		
		8.00%	
			2.00%
Liability		5.00%	
	4.00%		
			1.00%
Funding Center Spread		3.00%	
Net Interest Margin	6.00%		

Over the next year, interest rates rise by 200 basis points. Now, the bank, eager to eliminate future rate risk, issues a new four-year $1,000 CD at 8.5%. However, the four-year transfer rate is now 9.5%.

This table describes the components of the interest margin for the bank after one year:

Components of Interest Rate Margin for the Bank After One Year

Income Statement Component	Rate	Transfer Rate	Spread
Asset	10.00%		
		8.00%	
			2.00%
Liability		9.50%	
	8.50%		
			1.00%

Income Statement Component	Rate	Transfer Rate	Spread
Funding Center Spread		-1.50%	
Net Interest Margin	1.50%		

Although the bank is now perfectly matched from a current rate risk perspective (a four-year bullet loan funded by a four-year CD), it is losing 150 basis points at the funding center.

On day one the bank took a rate bet by funding short. The bet was that one year from the loan origination date, the bank would be able to raise four-year funds at less than the cost of funding the original five-year loan, or 8%. Since the four year transfer rate on day one was 7%, when interest rates went up by 200 basis points the bank got badly hit.

Although the net interest margin of the bank is still 150 basis points, the bank could have locked in a 300 basis point net interest margin for five years on day one if it had not taken a rate bet by issuing a five-year CD.

The loss of 150 basis points on the $1,000 loan is a result of the embedded rate risk taken by the bank. The bank can do nothing to eliminate embedded rate risk, except wait.

Measuring Current and Embedded Rate Risks

Even though nothing can be done about embedded rate risk, it is important to identify the impact of embedded rate risk for planning ahead.

For example, if a bank had a large profit in the funding center owing to embedded rate risk, and was unaware of this, it can be lulled into a false sense of security. That bank might be surprised when this source of profit evaporates.

Conversely, if a bank is experiencing a large loss in the funding center due to embedded rate risk, and it is able to measure it, the bank might choose to wait it out rather than taking drastic and immediate actions.

Measuring Current Rate Risk

You can measure current rate risk by transfer pricing your entire balance sheet as if it were originated today. Everything should be transfer priced based upon its remaining term. Under this method, a five-year CD with one year until maturity would receive the same transfer rate as a three-year CD with one year left.

Measuring Embedded Rate Risk

The total rate risk profit is made up of Embedded Rate Risk and Current Rate Risk.

Embedded Rate Risk = Total Rate Risk Result - Current Rate Risk Result

Related Topics

Overview of Transfer Pricing, page 1-1

2

Transfer Pricing Option Costs

This chapter describes how Oracle Transfer Pricing goes about calculating option costs. The chapter begins with an introduction to option costs and subsequently describes option cost theory and the calculation architecture.

This chapter covers the following topics:

- Overview of Transfer Pricing Option Costs
- Understanding the Option Cost Calculation Architecture
- Option Cost Theory
- Option Cost Model Usage Hints

Overview of Transfer Pricing Option Costs

The purpose of option cost calculations is to quantify the cost of optionality, in terms of a spread over the transfer rate, for a single instrument. The cash flows of an instrument with an optionality feature change under different interest rate environments and thus should be priced accordingly.

Consider a mortgage that may be prepaid by the borrower at any time without penalty. Here the lender has, in effect, granted the borrower an option to buy back the mortgage at par, even if interest rates have fallen in value. Thus, this option has a cost to the lender and should be priced accordingly.

Another example of an instrument with an optionality feature is an adjustable rate loan issued with rate caps (floors) which limit its maximum (minimum) periodic cash flows. These caps and floors constitute options.

When banks give such options to their borrowers, they raise the bank's cost of funding the loan and affect the underlying profit. Consequently, banks need to use the calculated cost of options given to their borrowers in conjunction with the transfer rate to analyze profitability.

Oracle Transfer Pricing uses the Monte Carlo technique to calculate the option cost. The application calculates and outputs two spreads, and the option cost is calculated

indirectly as a difference between these two spreads.

- Static spread
- Option-adjusted spread (OAS)

The option cost is derived as follows:

option cost = static spread - OAS

The static spread is equal to the margin, and the OAS to the risk-adjusted margin of an instrument. Therefore, the option cost quantifies the loss or gain due to risk.

You can calculate option costs using the Transfer Pricing Process rule. See: Transfer Pricing Process Rules, *Oracle Transfer Pricing User Guide*.

Understanding the Option Cost Calculation Architecture

This description of the option cost calculation architecture makes use of an example and assumes:

- The instrument, taken in the example, pays *K* cash flows, each occurring at the end of the month.
- Each month has the same duration in number of years, such as 1/12.
- The discount factor calculation does not use the approximation for small option adjusted spread.

Related Topics

Overview of Transfer Pricing Option Cost, page 2-1

Defining the Static and Option-Adjusted Spreads

You can define neither the static nor the option-adjusted spread directly, as they are solutions of two different equations. Therefore, the system solves a simplified version of the equations. The static spread is the value *ss* that solves the following equation:

$$MV = \sum_{k=1}^{K} \frac{CF(k)}{\prod_{j=0}^{k-1} (1 + f(j) + ss)^{\Delta t}}$$

Here:

- MV = market, book, or par value of the instrument

- CF(k) = cash flow occurring at the end of month k along the forward rate scenario
- f(j) = forward rate for month j
- Delta T = length (in years) of the compounding period; hard-coded to a month, such as 1/12

In the Monte Carlo methodology, the option-adjusted spread is the value OAS that solves the following equation:

$$MV = \sum_{\omega=1}^{N} \sum_{k=1}^{K} CF(k, \omega) D(k, \omega, OAS) / N$$

Here:

- N = total number of Monte Carlo scenarios
- CF(k, w) = cash flow occurring at the end of month *k* along scenario *w*
- D (k, w, OAS) = stochastic discount factor at the end of month *k* along scenario *w* for a particular OAS

> **Note:** Cash flows are calculated until maturity even if the instrument is adjustable. Otherwise the calculations would not catch the cost of caps or floors.
>
> In real calculations, the formula for the stochastic discount factor is simplified.

Related Topics

Understanding Option Cost Calculation Architecture, page 2-2

Option Cost Calculations Example

In this example, the transfer pricing yield curve is the Treasury curve. It is flat at 5%, which means that the forward rate is equal to 1%. This example uses only two Monte Carlo scenarios:

- **Up scenario:** One-year rate one year from now equal to 6%.
- **Down scenario:** One-year rate one year from now equal to 4%.

The average of these two stochastic rates is equal to 5%.

The instrument record is two year adjustable, paying yearly, with simple amortization.

Its rate is Treasury rate plus 2%, with a cap at 7.5%. Par value and market value are equal to $1.

For simplicity, this example assumes that the compounding period used for discounting is equal to a year, for example:

Delta t = 1

The static spread is the solution of the following equation:

$1 = [0.07 / (1 + 0.05 + SS)] + [(1 + 0.07) / (1 + 0.05 + SS)^2]$

The static spread is supposed to be equal to the margin. In this example:

static spread = coupon rate - forward rate =7%-5%=2%

Substituting this value (2% or .02) in the right side of the above equation yields:

$$\frac{0.07}{1+0.05+SS}+\frac{1+0.07}{(1+0.05+SS)^2}=$$

$$\frac{0.07}{1+0.07}+\frac{1+0.07}{(1+0.07)^2}=$$

$$\frac{0.07*(1+0.07)+1+0.07}{(1+0.07)^2}=$$

$$\frac{0.07+0.07^2+1+0.07}{(1+0.07)}=1$$

This is equal to par, which proves that the static spread is equal to the margin.

The OAS is the solution of the following equation:

$$1=\frac{0.07}{1+0.05+OAS}+$$

$$\frac{1}{2}\left[\frac{1+0.075}{(1+0.05+OAS)(1+0.06+OAS)}+\frac{1+0.06}{(1+0.05+OAS)(1+0.04+OAS)}\right]$$

By trial and error you get a value of 1.88%.

To summarize:

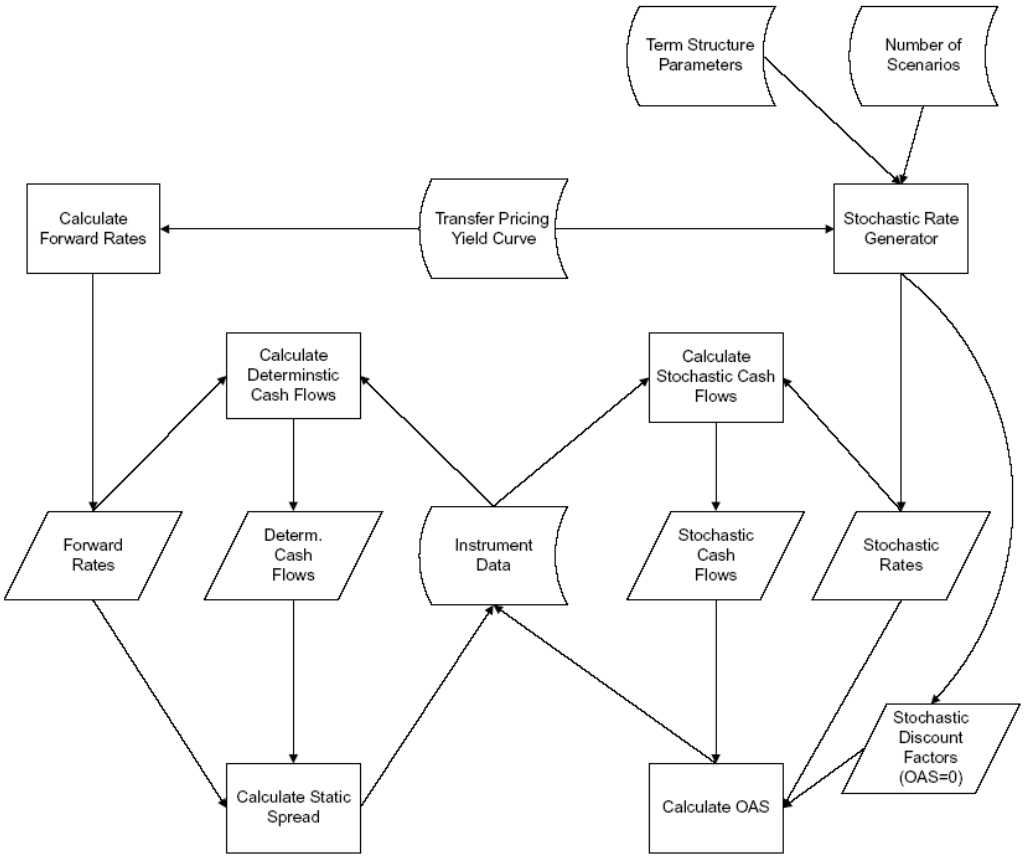
option cost = static spread - OAS = 2%-1.88% = 12 basis points

Related Topics

Understanding Option Cost Calculation Architecture, page 2-2

Option Cost Calculations Process Flow

The following graphic represents the option cost calculations process flow.



This exposition focuses only on the following steps:

- Calculating forward rates
- Calculating static spread
- Calculating OAS

Related Topics

Understanding Option Cost Calculation Architecture, page 2-2

Calculating Forward Rates

The cubic spline interpolation routine first calculates smoothed, continuously compounded zero-coupon yields *Y(j)* with maturity equal to the end of month *j*. The formula for the one-month annually compounded forward rate spanning month*j + 1* is:

$f_j = exp\ [\ (Y_j)\ (j + 1) - Y(j)\ j\ ] - 1$

Related Topics

Option Cost Calculations Process Flow, page 2-5

Calculating Static Spread

You can calculate the static spread using the Newton-Raphson algorithm. If Newton-Raphson algorithm does not converge, which can happen if cash flows alternate in sign, you can revert to a brute search algorithm. However, this algorithm is much slower.

You can control the convergence speed of the algorithm by adjusting the value of the variable OptionCostSpeedFactor. This variable is defined through a profile option, Option Cost Speed Factor.

The default value is equal to one. A lower speed factor provides more accurate results. In all experiments, a speed factor equal to one results in a maximum error (on the static spread and OAS), which is lower than half a basis point.

To recap the Newton-Raphson algorithm, let *x* be the static spread. At each iteration *m*, the function *F(m)* is defined by the following equation:

Equation A

$$F(m) = MV - \sum_{k=1}^{K} \frac{CF(k)}{\prod_{j=0}^{k-1} (1 + f(j) + x(m))^{\Delta t}}$$

The algorithm is:

$$m=0$$

INITIALIZE x(m)

REPEAT

$$m = m + 1$$

CALCULATE F (m), F'(m)

$$x(m+1) = x(m) - \frac{F(m)}{F'(m)}$$

$$\text{UNTIL} \left| \frac{F(m)}{MV} \right| < tol * \text{OptionCostSpeedFactor OR } m > \text{MaxIterations}$$

$$\text{IF } \left| \frac{F(m)}{MV} \right| > tol * \text{OptionCostSpeedFactor REVERT TO BRUTE SEARCH}$$

For performance reasons, the code utilizes a more complicated algorithm, albeit similar in spirit. This is the reason why the specific values for *tol* and *MaxIterations*, or details on the brute search are mentioned above.

Related Topics

Option Cost Calculations Process Flow, page 2-5

Calculating OAS

For fixed rate instruments, such as instruments having the same deterministic cash flows as the stochastic cash flows, the OAS is by definition equal to the static spread. This statement is true in the case of continuous compounding. For discrete compounding this approximation has a negligible impact on the accuracy of the results.

The OAS is also calculated with an optimized version of Newton-Raphson algorithm. See: Calculating Static Spread, page 2-6.

> **Note:** While calculating OAS, the following substitution is made in the Newton-Raphson method: OAS = x(m)

$$F(m) = MV - \sum_{\omega=1}^{N} \sum_{k=1}^{K} CF(k, \omega) D(k, \omega, x(m)) / N$$

Related Topics

Option Cost Calculations Process Flow, page 2-5

Option Cost Theory

According to the option cost theory, when you select the market value of an instrument to equate the discounted stream of cash flows, the static spread is equal to margin and the OAS to the risk-adjusted margin of the instrument.

This exposition of option cost theory assumes that you have good knowledge of no arbitrage theory, and requires you to note these assumptions and definitions:

- To acquire the instrument, the bank pays an initial amount *V(0)*, the current market value.
- The risk-free rate is denoted by*r(t).*
- The instrument receives a cash flow rate equal to *C(t),* with

  0 < = t < = T < = Maturity
- The bank reinvests the cash flows in a money market account which, with the instrument, comprises the *portfolio.*
- The *total return* on a portfolio is equal to the expected future value divided by the initial value of the investment.
- The *margin p* on a portfolio is the difference between the rate of return (used to calculate the total return) and the risk free rate *r.*
- The *risk-adjusted expected future value* of a portfolio is equal to its expected future value after hedging all diversifiable risks.
- The *total risk-adjusted return* of a portfolio is equal to the risk-adjusted expected future value divided by the initial value of the investment.
- The *risk-adjusted margin m* of a portfolio is the difference between the risk-adjusted rate of return (used to calculate the total risk-adjusted return) and the risk-free rate *r.*

More precisely,

$$\text{Total Return} = E\left[e^{\int_0^T r(t) + p\,dt} \right]$$

Equation B

$$\text{Total Risk} - \text{Adjusted Return} = \frac{E[S(T)]}{V(0)\left(1 + \text{corr}\left(e^{\int_0^T r(t)}, Z(T)\right)\right)}$$

Related Topics

Overview of Transfer Pricing Option Cost, page 2-1

Equivalence of the Option Adjusted Spread and Risk-Adjusted Margin

In a no-arbitrage economy with complete markets, the market value at time *t* of an instrument with cash flow rate *C(t)*is given by:

$$V(t) = E_t\left[\int_0^T C(t)e^{-\int_t^t r(s)ds}\,dt\right]$$

If expectation is taken with respect to the risk-neutral measure, the expected change in value is given by:

$$E_t[dV] = E_t[\lim_{\Delta \to 0} \frac{V(t+\Delta) - V(t)}{\Delta t}] = E_t[\lim_{\Delta \to 0} \frac{E_{t+\Delta}[V(t+\Delta)] - V(t)}{\Delta}] =$$

$$E_t[\lim_{\Delta \to 0} \frac{\int_{t+\Delta}^{T} C(u) e^{-\int_{t+\Delta}^{u} x(s)ds} du - \int_{t}^{T} C(u) e^{-\int_{t}^{u} x(s)ds} du}{\Delta}] = E_t[\frac{d}{dt} \int_{t}^{T} C(u) e^{-\int_{t}^{u} x(s)ds}] =$$

$$E_t[r(t)V(t) - C(t)]$$

The variation in value is, therefore, equal to the expected value of the change *dV* plus the change in value of a martingale *M* in the risk-neutral measure:

$dV(t) = E_t[dV] + dM = rVdt - Cdt + dM$

If *I* is the market value of the money market account in which cash flows are reinvested then:

$$I(T) = \int_{0}^{T} C e^{\int_{t}^{T} r(s)ds} dt$$

Note that unlike *V*, this is a process of finite variation. By Ito's lemma:

$dI = rIdt + Cdt$

Let *S*be the market value of a portfolio composed of the instrument plus the money market account. We have:

$dS = dV + dI = rSdt + dM$

$S(0) = V(0)$

In other words, the portfolio, and not the instrument, earns the risk-free rate of return. An alternate representation of this process is:

$dS/S = rdt + dN$

Here *N* is another martingale in the risk-neutral measure. The expected value of the portfolio is then:

$$E(S[T]) = V(0)E\left[e^{\int_{0}^{T} r(t)dt + N(T) - \frac{1}{2} < N,\ N >_T}\right]$$

Here, <*N, N*>is the quadratic variation of *N*. This is equivalent to:

$$E[S(T)] = V(0)E\left[e^{\int_0^T r(t)dt + N(T) - \frac{1}{2} < N,\ N >_T}\right]$$

To define the martingale:

$Z = e^{N-(1/2)<N,N>}$

This represents the relative risk of the portfolio with respect to the standard money market account, that is, the account where only an initial investment of *V(0)* is made. Then

$$E[S(T)] = V(0)E\left[e^{\int_0^T r(t)dt}\right]\left(1 + \text{corr}\left(e^{\int_0^T r(t)dt}, Z(T)\right)\right)$$

In other words, the expected future value of the portfolio is equal to the expected future value of the money market account adjusted by the correlation between the standard money market account and the relative risk. Assuming complete and efficient markets, banks can fully hedge their balance sheet against this relative risk, which should be neglected to calculate the contribution of a particular portfolio to the profitability of the balance sheet. Therefore:

$$\textbf{Risk} - \textbf{Adjusted Expected Future Value} = \frac{E[S(T)]}{\left(1 + \text{corr}\left(e^{\int_0^T r(t)}, Z(T)\right)\right)}$$

In this example, the risk-adjusted rate of return of the bank on its portfolio is equal to the risk-free rate of return.

Now suppose that another instrument offers cash flows *C' > C*.

Assuming complete and efficient markets, the market value of this instrument is:

$$\text{Total Risk} - \text{Adjusted Return} = \frac{E[S(T)]}{V(0)\left(1 + \text{corr}\left(e^{\int_0^T r(t)}, Z(T)\right)\right)}$$

$$V'(0) = E\left[\int_0^T C'(t) e^{-\int_0^t r(s)ds} dt\right] > V(0)$$

The value of the corresponding portfolio is denoted by $S' > S$.

By analogy with the previous development, we have:

$$\text{Total Risk} - \text{Adjusted Return} = \frac{E[S'(T)]}{V'(0)\left(1 + \text{corr}\left(e^{\int_0^T r(t)}, Z'(T)\right)\right)}$$

Again, the risk-adjusted rate of return of the bank on its portfolio is equal to the risk-free rate of return. Suppose now that markets are incomplete and inefficient. The bank pays the value *V(0)* and receives cash flows equal to *C'*. We have:

$$\text{Total Risk} - \text{Adjusted Return} = \frac{E[S'(T)]}{V(0)\left(1 + \text{corr}\left(e^{\int_0^T r(t)}, Z'(T)\right)\right)}$$

By definition of the total risk-adjusted return for Equation B, page 2-8, we have:

Equation C

$$E[S(T)] = V(0)E\left[e^{\int_0^T r(t)+mdt}\right]\left(1 + \text{corr}\left(e^{\int_0^T r(t)+mdt}, Z'\right)\right)$$

Therefore, by analogy with the previous development,

$dS' = (r + m)S' + dM$

$S(0) = V(0)$

This can be decomposed into

Equation D

$dV' = (r + m)\, V'\, dt - C'\, dt + dM'$

Equation E

$dS' = dV' + dI'$

$dI' = rI'\, dt + C'\, dt$

Equation F

$V'(0) - V(0)$

The solution of Equation D, page 2-13 and Equation F, page 2-13 is:

$$V'(0) = V(0) = E\left[\int_0^T C'(t)e^{-\int_0^t r(s)+m\,ds}\,dt\right]$$

By the law of large numbers, Equation D, page 2-13 and Equation F, page 2-13 result in:

OAS = m

In other words, the OAS is equal to the risk-adjusted margin.

Related Topics

Option Cost Theory, page 2-8

Equivalence of the Static Spread and Margin

Static spread calculations are deterministic. Therefore, they are a special case of the equations in the previous section where all processes generally are equal to their expected value, and the margin *p* is substituted for the risk-adjusted margin *m*. The equivalent of Equation C, page 2-13 is then:

$$S'(T) = V(0)e^{\int_0^T f(t)+p\,dt}$$

Here *f* is the instantaneous forward rate.

The equivalent of Equation D, page 2-13 and Equation F, page 2-13 is then:

Equation G

dV′ = (r + m)V′ dt - C′ dt

Equation H

dI′ = rI′ dt + C′ dt

dS′ = dV′ + dI′

Equation I

V′(0) = V(0)

The solution of Equation G, page 2-14 and Equation I, page 2-14 is:

Equation J

$$V'(0) = V(0) = \int_0^T C'(t) e^{-\int_0^t f(s) + p\,ds} \, dt$$

Comparing Equation J, page 2-14 and Equation A, page 2-6:

$ss = p$

In other words, the static spread is equal to the margin.

Related Topics

Option Cost Theory, page 2-8

Option Cost Model Usage Hints

The option cost calculation model is flexible and you can calibrate the calculations to your needs. See:

- Nonunicity of the Static Spread, page 2-15
- Calibrating the Accuracy of Option Cost Calculations, page 2-16

Related Topics

Overview of Transfer Pricing Option Cost, page 2-1

Nonunicity of the Static Spread

Nonunicity of the static spread means that sometimes more than one value can solve the static-spread equation. However, such cases extremely rare.

Take an instrument with a market value of $0.445495 for example. Suppose the instrument has two cash flows. The following table shows the value of the cash flows and the corresponding discount factors (assuming a static spread of zero).

Value of the Cash Flows and the Corresponding Discount Factors

Events	Time	Cash Flow Value	Discount Factor (static spread = 0)
First Event	1	1	0.9
Second Event	2	-0.505025	0.8

The continuously compounded static spread solves the following equation:

*0.9 Exp (-ss) -0.8 * 0.505025 Exp (-2ss) -0.445494 = 0*

There are two possible solutions for the static spread:

- *static spread = 0.19%*
- *static spread = $1.81*

Related Topics

Option Cost Model Usage Hints, page 2-15

Calibrating the Accuracy of Option Cost Calculations

If you desire a better numerical precision than the default precision, you can take two actions:

- Decrease the speed factor. See: Calculating Static Spread, page 2-6.
- Increase the number of Monte Carlo scenarios.

Both actions increase the calculation time.

Related Topics

Option Cost Model Usage Hints, page 2-15

3

The Ledger Migration Process

This chapter discusses the process for generating credits or charges, for funds provided or used, and their migration to the Management Ledger table. The chapter provides a detailed description of how the information required for generating these credits or charges originates through transfer rates and option cost processing from the Customer Account tables and how the results are inserted into the Management Ledger table.

This chapter covers the following topics:

- Overview of the Ledger Migration Process
- Understanding Ledger Migration
- Requirements for Successful Ledger Migration
- Example of Transfer Rate Ledger Migration
- Ledger Migration of Option Costs

Overview of the Ledger Migration Process

Ledger migration is the process of generating aggregated charges (expenses) and credits (revenues) for funds provided or used for a combination of dimensions. The information necessary to generate these charges and credits (through transfer rates and option cost processing) originates from the Customer Account tables and results are inserted into the Management Ledger table (FEM_BALANCES). Transfer pricing charge and credit information provides the basis for measuring net interest income contribution for a group of products, organizational units, or a combination of other dimensions, and is available for use in further calculations of profitability and risk.

> **Note:** The Management Ledger table is also known as the FEM_BALANCES and Customer Account tables are also known as Instrument tables.

Oracle Transfer Pricing provides great flexibility in the ledger migration process and in the generation of corresponding charges, credits, and option costs. Users can specify

ledger migration for a combination of an extended list of dimensions, including up to 10 user-defined dimensions. This feature provides flexibility to users who are also using Oracle Profitability Manager for profitability reporting across organizational, product, channel, geography, or other user-defined dimensions.

In addition, Oracle Transfer Pricing provides multi-currency support that allows you to generate charges or credits for funds based on entered and functional currency.

You can choose to migrate either the transfer rate or the option costs, or both, on the Transfer Pricing Process rule run page. See: Transfer Pricing Process Rules, *Oracle Transfer Pricing User Guide*.

Understanding Ledger Migration

To understand the process of creation of transfer rate, option cost, and charge/credit rows in the Management Ledger table (financial elements 170/172, 171/173, and 450/451/452/453, respectively), you need to make the following assumptions:

- All rows in the relevant Account tables have already been transfer-priced or assigned an option cost.
- All rows contain a valid rate in one of the following columns:
  - TRANSFER_RATE
  - TRAN_RATE_REM_TERM
  - CUR_OAS
  - HISTORIC_OAS
- Average Balance (financial element 140) information has been loaded into the Management Ledger table with dimensionality that matches the account table data being migrated.

It is common for account table transfer pricing, option cost calculation, and transfer pricing (transfer rate and option cost) ledger migration to be executed consecutively in a single process.

This exposition describes the mechanics which occur just after the Account tables transfer pricing or option cost calculations have been completed successfully and just before transfer rate or option cost ledger migration starts. For example, the mechanics that occur just after Account tables have been populated with valid transfer rates and just before the weighted average transfer rate (WATR) and the Charge/Credit rows in the Management Ledger table have been updated.

The ledger migration of option costs works on the same lines as transfer rate migration. However, there are certain differences. See: Ledger Migration of Option Costs, page 3-16.

Transfer Rate and Option Cost Calculation

The Oracle Transfer Pricing engine calculates and writes balance-weighted average rates to the Management Ledger table, using current balance to perform the weighting process. The financial elements that the engine uses to write the weighted rates to the FEM_BALANCES are as follows:

- 170 Average Transfer Rate
- 172 Average Rem Term Transfer Rate
- 171 Historic Option Cost
- 173 Current Option Cost

Charge/Credit Generation

In addition to the calculation of the weighted average rate values at the combination of the Organizational Unit and the Line Item dimensions, charge/credit generation involves the following steps:

- Aggregation of the corresponding average balance records from the Management Ledger table for each Org Unit/Line Item dimension combination
- Multiplication of the average balance from the Management Ledger by the weighted average rates
- Application of an accrual factor to de-annualize the amount

Oracle Transfer Pricing then writes the result as dollar charges/credits to the Management Ledger table using the following financial elements:

- 450 Charge/Credit
- 451 Historic Option Cost Charge/Credit
- 452 Charge/Credit Rem Term
- 453 Current Option Cost Charge/Credit

Direct Transfer Pricing of Ledger Balances

Oracle Transfer Pricing allows users to calculate transfer rates for ledger average balances that do not have corresponding Account table records using the following transfer pricing methodologies:

- Moving Averages

- Spread from Interest Rate Code
- Redemption Curve
- Unpriced Account

Oracle Transfer Pricing also generates records in the Management Ledger table which are posted to the organizational unit (Org Unit) designated as the Transfer Pricing Offset Unit (typically a special Treasury Unit). During this process, an offset charge or credit amount is calculated for each normal charge/credit posted at the intersection of Company/Cost Center and Line Item dimensions in the processes outlined above.

The sum of the Org Unit charges and credits at the Line Item ID level is multiplied by -1 and posted to the offset Org Unit designated in the product setup for the Org Unit dimension. After this processing is complete, the total entity level charges and credits net to zero.

Ledger Migration and the Management Ledger Table

A thorough understanding of the ledger migration process requires familiarity with the Management Ledger table standards such as data signage, editing standards, and WATR and charge/credit rows.

Data Signage

The Management Ledger table supports variable data signs. You can load data into the Management Ledger table in the following three variations:

- **Absolute:** All account types are positive, and all contra accounts are negative.
- **GAAP or Standard:** The signs reflect standard accounting principles, with revenue, liability, and owners' equity as negative, and expenses and assets as positive.
- **Reverse of GAAP (Standard) or User Defined:** Revenue, liability, and owners equity are positive, and expenses and assets are negative.

Management Ledger Table Editing Standards

You should be extremely careful while editing the Management Ledger table directly. If you ever get unexpected results in the Management Ledger table after ledger migration, then review the data you have entered.

WATR and Charge/Credit Rows

The weighted average transfer rate (WATR) and the resulting charge/credit for funds are represented in the Management Ledger table by financial elements 170 and 450 respectively (or 172 and 452 if transfer pricing is done on a remaining term basis).

- **Financial Elements 170/172 (WATR):** If you select the Remaining Term calculation mode while defining the Transfer Pricing Process Rule, then the financial element generated is 172. Otherwise, it is 170. Only one 170/172 row should exist for a given combination of Company/Cost Center/Org and Line Item dimensions.

- **Financial Elements 450/452 (Charges/Credits for Funds):** If you select the Remaining Term calculation mode while defining the Transfer Pricing Process Rule, then the system generates financial element 452. If not, it would be Financial Element 450. Only one financial element, 450 or 452, should exist for a given combination of Company/Cost Center/Org Unit and Line Item dimensions.

Ledger Migration and the Virtual Memory Table

To calculate transfer rates at the Line Item dimension member level in the Management Ledger table, all rows in the Account tables must be accumulated to arrive at the weighted average transfer rate (WATR) for each member.

All data used in the ledger migration process passes through a table, called the Virtual Memory table (VMT), built in the memory. This table exists only during the ledger migration process and the information is never written to disk, and thus it cannot be examined for problem-solving purposes. Understanding the operation of the VMT, however, is crucial to understanding the ledger migration process.

The VMT comprises the following three types of columns:

- Company/Cost Center/Org Unit and Line Item columns, which uniquely identify each row

- Balance and WATR columns to hold data accumulated from the Customer Account tables

- Balance and WATR columns to hold data accumulated from Management Ledger table and Customer Account table calculations

Requirements for Successful Ledger Migration

Successful ledger migration of transfer pricing results requires correct configuration of the following parameters:

- Transfer Pricing Parameters, page 3-6

- Dimensions, page 3-6

- Entered and Local Currency, page 3-7

- Transfer Pricing Rule, page 3-8

- Conditions and Table Selection, page 3-8
- Ledger Migration and Line Item Dimension Set Up, page 3-8
- Offset Org Unit, page 3-8
- Transfer Pricing Processing Rule, page 3-9
- Calculation Mode, page 3-9

Together these parameters determine the way transfer rate and option cost calculations are carried out for every account in the organization.

Transfer Pricing Parameters

You need to configure the following transfer-pricing parameters:

- **Effective Date:** Must match the period for which you are trying to migrate transfer rates and option costs. The default effective date is defined in user preferences.
- **Charge/Credit Accrual Factor:** Select the Charge/Credit Accrual Factor on the Transfer Pricing Process Rule page or, alternatively, define the Accrual Factor as an attribute for each Line Item dimension member. In case no selection is made, an Accrual Factor of 30/360 is applied.

Dimensions

To be eligible for inclusion in the ledger migration process, a dimension must exist and be actively populated with dimension values in both the Account tables and in the Management Ledger table.

Given below is a list of dimensions available for inclusion in the ledger migration process:

- **Mandatory Dimensions:**
  - LINE_ITEM_ID,
  - CURRENCY_CODE,
  - LEDGER_ID
  - CAL_PERIOD_ID
  - DATASET_CODE
- **Other Available Dimensions:**

- CHANNEL_ID
- COMPANY_COST_CENTER_ORG_ID
- NATURAL_ACCOUNT_ID
- PRODUCT_ID
- Up to 10 user-defined dimensions

Entered and Local Currency

Oracle Transfer Pricing provides you with the option of performing ledger migration and writing charges and credits in the entered or local currency, designated in the CURRENCY_CODE column, or in the functional currency.

Source of Currency and Exchange Rate Information

Oracle Transfer Pricing shares currency and exchange rate information with the Oracle General Ledger. Currency rate information can be viewed or updated from the Oracle Transfer Pricing/General Ledger Currency Rates screen. Information on active and valid currencies (enabled currencies), functional currency, currency relationships, and exchange rates are defined at the Set of Books (LEDGER_ID) level.

Ledger migration should only be performed for currencies that have been activated or enabled.

If currency code values that have not been activated are discovered in the ledger migration process, users are presented with a warning stating the same, and the ledger migration process skips records with those values.

Calculation of Functional Currency Values

To calculate and write charge/credit values expressed in functional currency to the Management Ledger table, a typical situation in multi-currency implementations, you should take the following steps:

1. Choose between entered or functional ledger migration while defining the Transfer Pricing Process Rule.
2. Derive charge/credit amounts in the entered or local currency first, using transfer rate and balance information expressed in those currencies, and then convert the calculated charge/credit values for the calendar period to the functional currency.
3. Assume the last date associated with the calendar period as the basis for ledger migration, and generally use currency exchange rates corresponding to that date to perform conversions to functional currency for charges and credits written to the Management Ledger table.

4. Use the following algorithm for exchange rate access:

    - If exchange rate exists, use the rate for the last day of calendar period being processed.
    - If no exchange rate exists for last day of calendar period being processed, use the latest exchange rate available in the rates table for the period being processed.
    - If no exchange rate exists for the period being processed, use an exchange rate value of 1.

Transfer Pricing Rule

The Transfer Pricing Rule is used to define the transfer pricing and option cost methodology for each product dimension. While defining transfer pricing methodologies, ensure that all required supporting data for the method actually exists.

For example, if the selected method is Spread from Interest Rate Code, ensure that the corresponding yield curve has been properly defined and has been populated with rates.

Conditions and Table Selection

Calculating and migrating transfer rates and option costs for an entire product portfolio can be a time-consuming process. Conditions and Table Selection allow you to reduce the ledger migration time as follows:

- **Conditions:** Also known as Data Filter IDs, *Conditions* allow you to transfer price or migrate to ledger a subset of your portfolio.
- **Table Selection:** This feature gives you the option of selecting the Account tables for ledger migration during a particular Transfer Pricing Process Rule run.

Ledger Migration and Line Item Dimension Set Up

The Line Item dimension contains an attribute, monthly accrual basis, that is used to designate the accrual factor for a particular product used in calculating the charge or credit for funds. This attribute should be defined for all products when the user wishes to base charge and credit calculations on product-specific accrual factors rather than a single process-specific accrual factor defined at the Transfer Pricing Process Rule level.

Offset Org Unit

During ledger migration, Oracle Transfer Pricing generates records in the Management Ledger table that are posted to the Company/Cost Center/Org Unit designated as the Transfer Pricing Offset Unit. During this process, an offset charge or credit amount is

calculated for each normal charge/credit posted at the intersection of Company/Cost Center/Org Unit and Line Item.

The Company/Cost Center/Org dimension contains an attribute, Offset Org, that is used to define the funding center that receives the offset entries from all charge and credit postings.

Transfer Pricing Process Rule

The Transfer Pricing Process Rule acts as a container for all the ledger migration parameters and submits them to the Transfer Pricing engine as a processing job. A Transfer Pricing Process rule typically contains the following ledger migration specifications:

- The dimensions that you want to include in the ledger migration process.
- The tables that are to undergo transfer pricing or option cost calculations.
- Conditions that are to be applied to the rows in each table.
- Transfer pricing or prepayment methodologies to be used.
- Option cost calculation parameters.
- Charge/credit accrual basis to be used.

Calculation Mode

The choice of calculation mode, on the Transfer Pricing Process Rule run page, not only affects the transfer rate and option cost calculation processes, but also the migration process. It determines the results that will be migrated to the Management Ledger table.

If the calculation mode is set to Standard then the following results are used in migration:

- Transfer_Rate
- Historic Option Cost (Historic_Static_Spread - Historic_OAS)

Consequently, the transfer pricing engine generates results for the following financial elements:

- 170 Average Transfer Rate
- 171 Historic Option Cost
- 450 Charge/Credit
- 451 Historic Option Cost Charge/Credit

If the calculation mode is set to Remaining Term, then migration process uses the following result columns:

- Tran_Rate_Rem_Term
- Current Option Cost (Cur_Static_Spread - CUR_OAS)

Consequently, the transfer pricing engine generates results for following financial elements:

- 172 Average Rem Term Transfer Rate
- 173 Current Option Cost
- 452 Charge/Credit Rem Term
- 453 Current Option Cost Charge/Credit

Example of Transfer Rate Ledger Migration

Ledger migration requires you to select, among others, the following options while creating and executing the Transfer Pricing Process Rule:

- Select both the Account tables and the Management Ledger table as the tables to be processed.
- Select both the transfer rate calculation and the ledger migration processing options. Selecting the transfer rate calculation option leads to the generation of transfer rates for all the records in the Customer Account tables and for those records in the Management Ledger table for which you define a transfer rate with a ledger source type. And, selecting the ledger migration processing option instructs the application to gather balances and transfer rates information, generate credits and charges for funds, and output the results to the Management Ledger table.

Oracle Transfer Pricing allows you to include multiple dimensions in the ledger migration process. However, to keep the exposition simple, this example assumes that only two dimensions, the Company Cost Center/Org Unit dimension and the Line Item dimension, are selected to generate results.

The following table displays the Customer Account table data for this example.

Customer Account Tables (FEM_MORTGAGES)

COMPANY_COST_CENTER_ORG_ID	LINE_ITEM_ID	CUR_BOOK_BAL	TRANSFER_RATE
1	3	100	4.00

COMPANY_COST_CENTER_ORG_ID	LINE_ITEM_ID	CUR_BOOK_BAL	TRANSFER_RATE
1	4	125	4.50
1	5	200	3.00
1	3	200	3.00

The following table displays the pre-migration data in the Management Ledger table used in the example.

Management Ledger Table (FEM_BALANCES)

COMPANY_COST_CENTER_ORG_ID	LINE_ITEM_ID	FINANCIAL_ELEM_ID	XTD_BALANCE_ED
1	3	140	250.00
1	4	140	200.00
1	5	140	100.00
1	10	140	200.00
1	100	140	990.00

As you compare the Customer Account tables and the Management Ledger table data, notice the following:

- Line Item IDs 3, 4, and 5 match in both tables. These Product IDs represent the simplest case of ledger migration.
- Line Item ID 10 does not exist in the Customer Account tables. This example assumes that it is a ledger-only account that is transfer priced directly using an acceptable Management Ledger Table data source-only method (part of the assumption definition in the Transfer Pricing Rule).
- Line Item ID 100 does not exist in the Account tables. This example assumes that it is a ledger-only account that will be transfer priced using Unpriced Account Methodology, based on Product IDs 4, 5, and 10. (This transfer pricing method is defined in the Transfer Pricing Rule.)

The ledger migration process essentially comprises the following two broad phases:

- Account Tables Accumulation, page 3-12
- Management Ledger Table Processing, page 3-12

However, this example with a view to illustrating the operation of the ledger migration process in general and that of the virtual memory table (VMT) in particular demonstrates the following possible variations of the ledger migration process and special cases:

- Transfer Pricing Accounts with Ledger-Only Data Source, page 3-13
- Transfer Pricing Unpriced Accounts, page 3-14
- Ledger Migration of Transfer Rates Under Remaining Term Calculation Mode, page 3-16

Account Tables Accumulation

The first operation in the ledger migration process is to accumulate all individual detail rows from the Customer Account tables into a single row for each unique combination of Company/Cost Center/Org Unit and Line Item dimensions in the Virtual Memory Table (VMT).

In this example, Bal_x_TfrRate for Line Item 3 is calculated as follows:

*(100 * 4.00) + (200 * 3.00) = 1,000.00 = Bal_x_TfrRate*

The following table represents the VMT after Account table accumulation has taken place.

VMT Post Account Table Accumulation

COMPANY_ COST CENTER_ ORG_ID	LINE_ITEM_ID	Bal	Bal_x_TfrRate	LSBal_x_TfrRate
1	3	300.00	1,000.00	
1	4	125.00	562.50	
1	5	200.00	600.00	

Management Ledger Table Processing

The first step in the ledger migration process with respect to the Management Ledger table is to clear all the information stored in the table with financial elements 170 and 450 (172 and 452 if remaining term pricing is being used) for the particular combination

of dimensions being used in the process.

The next step is Management Ledger table accumulation: the Virtual Memory Table (VMT) is populated with the balance information stored in the Management Ledger Table.

The following table represents the VMT after Management Ledger Table accumulation has taken place.

VMT Post Management Ledger Table Accumulation

COMPANY_ COST CENTER_ ORG_ID	LINE_ITEM__ID	Bal	Bal_x_TfrRate	LSBal	LSBal_x_TfrRate
1	3	300.00	1,000.00	250	
1	4	125.00	562.50	200	
1	5	200.00	600.00	100	

Management Ledger table processing involves the calculation of the weighted average transfer rate (WATR). The WATR is calculated by prorating the WATR by the ratio between the Account tables and the Management Ledger table balances as follows:

*(Bal x TfrRate / Bal) * LSBal = LSBal x TfrRate*

For example, the WATR for Line Item 3 is calculated as follows:

*(1,000.00 / 300.00) * 250.00 = 833.33*

The following table represents the VMT after WATR calculation has taken place.

VMT Post WATR Calculation

COMPANY_ COST CENTER_ ORG_ID	LINE_ITEM__ID	Bal	Bal_x_TfrRate	LSBal	LSBal_x_TfrRate
1	3	300.00	1,000.00	250.00	833.33
1	4	125.00	562.50	200.00	900.00
1	5	200.00	600.00	100.00	300.00

Transfer Pricing Accounts with Ledger-Only Data Source

At this stage, all rows in the Management Ledger table that relate (directly or indirectly)

to rows in the Customer Account tables are accumulated into the VMT. However, the accumulation process still needs to deal with account types that are transfer priced using Ledger as the data source (as specified in the Transfer Pricing Rule).

In this example, Line Item 10 is a Direct Transfer Price product with a Management Ledger balance of 200.00.

The following table represents a VMT with a direct transfer price product.

VMT with a Direct Transfer Price Product

COMPANY_ COST CENTER_ ORG_ID	LINE_ITEM_ID	Bal	Bal_x_TfrRate	LSBal	LSBal_x_TfrRate
1	3	300.00	1,000.00	250.00	833.33
1	4	125.00	562.50	200.00	900.00
1	5	200.00	600.00	100.00	300.00
1	10			200.00	1,000.00

Transfer Pricing Unpriced Accounts

Accounts using the Unpriced Account method are a special case of direct transfer pricing in the Management Ledger table. The Unpriced Account transfer pricing methodology uses the WATR from other accounts to derive a WATR for the unpriced account. This is accomplished by averaging the WATR for the component accounts, weighted by their relative LS Balances.

In this example, Line Item 100 is an unpriced account that is transfer priced based on Line Item 4, 5, and 10. First, as shown in the following table, a new row is added to the VMT and populated with the balance stored in the Management Ledger table.

VMT with an New Row Displaying Management Ledger Table Balance

COMPANY_ COST CENTER_ ORG_ID	LINE_ITEM_ID	Bal	Bal_x_TfrRate	LSBal	LSBal_x_TfrRate
1	3	300.00	1,000.00	250.00	833.33
1	4	125.00	562.50	200.00	900.00
1	5	200.00	600.00	100.00	300.00

COMPANY_ COST CENTER_ ORG_ID	LINE_ITEM_ID	Bal	Bal_x_TfrRate	LSBal	LSBal_x_TfrRate
1	10			200.00	1,000.00
1	100			990.00	

Then, the WATR for Line Item 100 is calculated by computing the weighted average of the WATRs of Line Items 4, 5, and 10. The WATR for Line Item 100 is calculated as follows:

(900 + 300 + 1,000)/(200 + 100 + 200)

= 4.4

The VMT is then updated with the standard form of WATR

*(990.00 * 4.4) = 4,356.00 = LSBal_x_TfrRate*

The following table represents the VMT after the unpriced account has been transfer priced.

VMT Displaying the WATR of Unpriced Account

COMPANY_ COST CENTER_ ORG_ID	LINE_ITEM_ID	CDBal	CDBal_x_TfrRate	LSBal	LSBal_x_TfrRate
1	3	300.00	1,000.00	250.00	833.33
1	4	125.00	562.50	200.00	900.00
1	5	200.00	600.00	100.00	300.00
1	10	-	-	200.00	1,000.00
1	100	-	-	990.00	4,356.00

Calculation of Overall WATR (Financial Element 170)

Once all the Customer Account tables and the Management Ledger table information has been accumulated in the VMT, the overall WATR can be calculated for each Org Unit/Line Item dimension combination and posted to the Management Ledger table. The WATR is simply the sum of all component WATRs (represented in the VMT as LSBal x TfrRate).

For example, WATR is calculated as follows:

833.33 + 900.00 + 300.00 + 1,000.00 + 4,356.00 = 7,089.33= WATR

Generation of Charge/Credit for Funds (Financial Element 450)

Once the overall WATR is known, the charge/credit for funds in any period is given by the formula:

*WATR * Balance * Accrual Factor = Charge/Credit for Funds*

As Oracle Transfer Pricing stores WATR as WATR * Balance, this reduces to:

*WATR * Accrual Factor = Charge/Credit for Funds*

For example, Charge/Credit for Funds is calculated as follows:

*7,089.33 * (30/360) = 590.77 = Charge/Credit for Funds*

Ledger Migration of Transfer Rates Under Remaining Term Calculation Mode

The ledger migration process is identical under the Remaining Term calculation mode except that Financial Elements 452 and 172 are substituted for 450 and 170 respectively. Note that under the Remaining Term calculation mode, the transfer rate source in the Account tables is Tran_Rate_Rem_Term.

Ledger Migration of Option Costs

Ledger migration of option costs is similar to that of the transfer rate. However, there are no steps for calculating option costs directly on the Management Ledger Table, because the calculation of option cost is a cash-flow based method that requires the Customer Account table data.

Normally, option cost is represented in the Account table record as the difference between two columns, HISTORIC_ STATIC_SPREAD and HISTORIC_OAS (Option Adjusted Spread) and is expressed as a rate, in percent.

Option Cost = HISTORIC_STATIC_SPREAD - HISTORIC_OAS

If option cost ledger migration is specified in the Transfer Pricing Process Rule, option cost is accumulated in the Virtual Memory table (VMT) and written to the Management Ledger table as Financial Element 171, Average Historical Option Cost. The corresponding charge/credit for Funds is written as Financial Element 451, Historical Option Cost Charge/Credit.

Option Cost Ledger Migration under the Remaining Term Calculation Mode

The option cost ledger migration process is nearly identical under the Remaining Term calculation mode, except that Financial Elements 453 and 173 are substituted for 451 and 171, respectively.

Note that under the Remaining Term calculation mode, the option cost source in the Account tables is the difference between the Cur_Static_Spread and Cur_OAS columns.

4

Rate Conversion

This chapter describes how Oracle Transfer Pricing translates interest rates from their initial formats into formats that can be used by the application.

This chapter covers the following topics:

- Overview of Rate Conversion
- Characteristics of Interest Rates Codes
- Rate Format Usage
- Rate Conversion Algorithms

Overview of Rate Conversion

Interest rates are available in a variety of formats. In Oracle Transfer Pricing, interest rates are used for multiple purposes, each requiring a specific rate format. The application must apply transformation formulas to translate the interest rates from their initial formats into formats that can be used by Oracle Transfer Pricing.

Related Topics

Interest Rate Codes, *Oracle Transfer Pricing User Guide*

Characteristics of Interest Rates Codes

The following characteristics define an interest rate code:

- Accrual basis
- Compound basis
- Rate format

The accrual basis can be:

- 30/365
- 30/Actual
- Actual/Actual
- Actual/365
- Actual/360

The compound basis can be:

- Monthly
- Semiannual
- Annual
- Simple

The rate format can be:

- Zero-coupon yield
- Yield-to-maturity
- Discount factor

Discount factor is used only internally and cannot be specified as an input rate format in Oracle Transfer Pricing. It is well known that for bonds issued at par with payment frequency equal to the compound basis, the yield-to-maturity at origination or par yield is equal to the coupon.

There are several definitions of yield-to-maturity. Oracle Transfer Pricing does not use the unconventional *true yield* definition of Stigum but prefers instead the Wall Street convention.

Related Topics

Overview of Rate Conversion, page 4-1

Rate Format Usage

A few typical instances of rate format usage in Oracle Transfer Pricing are as follows:

- Monte Carlo Rate Path Generation, page 4-3
- Rate Index Calculation from Monte Carlo Rate Paths, page 4-3

- Use of Transfer Pricing Methods, page 4-3

Related Topics

Overview of Rate Conversion, page 4-1

Monte Carlo Rate Path Generation

The Monte Carlo Rate Path Generation process requires annually compounded Actual/Actual zero-coupon yield as the input. If the input IRC format is anything other than zero-coupon annual yield, a conversion process is applied.

Stochastic rates output from Monte Carlo are also annually compounded Actual/Actual zero-coupon yields.

Related Topics

Rate Format Usage, page 4-2

Rate Index Calculation from Monte Carlo Rate Paths

The Rate Index rule formulas are applied to the yields forecasted from the valuation curve in the Monte Carlo process. A formula for each additional IRC must exist in the Rate Index rule. The formulas are applied during processing when one of the additional IRCs is required for repricing individual instrument records.

Related Topics

Rate Format Usage, page 4-2

Use in Transfer Pricing Methods

A description of rate format usage in cash-flow based and noncash flow transfer pricing methods is given below.

Noncash Flow Methods

The noncash flow based transfer pricing methods use historical IRC data pulled directly from the historical rates tables. For these methods, the format of the IRC used as the transfer pricing yield curve is assumed to be a yield-to-maturity.

Cash Flow Methods

There are three cash flow methods:

- **Weighted Average Term:** This method calculates the cash flows over the funding period, treating the next repricing date as a maturity date. The cash flows are

discounted by the current net rate. The discounted cash flow at each payment/maturity is used as the weighting factor together with a time weight for the rate from the transfer pricing yield curve. The term from the origination to the cash flow date is used as the term for lookup to the transfer pricing yield curve.

For this method, the transfer pricing yield curve is assumed to be the proper rate format. No adjustments are made to the current net rate or the transfer pricing yield curve.

- **Duration:** This method calculates the duration by taking the cash flows over the funding period and calculating duration for the series of cash flows. The current net rate is used as the discount rate. The duration of the cash flows is used as the term for lookup to the transfer pricing yield curve.

  For this method, the transfer pricing yield curve is assumed to be the proper format. No adjustments are made to the current net rate.

- **Zero Coupon:** This method must calculate discount factors for the transfer pricing yield curve. If the transfer pricing yield curve is stored as yield-to-maturity rates, the rates must first be translated into zero coupon yields so that the discount factor can be calculated from them. If the transfer pricing yield curve is already in yield format, then discount factors can be calculated directly from the rates.

Related Topics

Rate Format Usage, page 4-2

Rate Conversion Algorithms

Accrual basis or compounding basis conversions are quite straightforward. However, rate format conversions from zero-coupon yield to yield-to-maturity and vice versa, are difficult.

The following table describes terminology used in rate conversion algorithms used by Oracle Transfer Pricing.

Terminology used in Rate Conversion Algorithms

Symbol	Name	Notes
$AI(T_i)$	Accrued interest for the i-th security	
$C(T_i)$	Coupon value of the i-th security	This is the true dollar value of the cash flow (not annualized)

Symbol	Name	Notes
m	Compounding frequency (per year)	Possible values are: • 12 (monthly) • 2 (semi-annual) • 1 (annual) • 0 (simple)
n_i	Number of full compounding periods from As of Date up to term T_i	
P(T)	Discount factor with term T_i	Value of a zero-coupon bond T_i and par = $1
r	Total number of securities	
T_i	Term in Act/Act years of the i-th security	Sorted in ascending order
L(i,k)	Time in Act/Act years of the start of the k-th (k=0... n_i) compounding period for security i	
w_i	Residual number of compounding periods between the current date and the next compounding date for i-th security	
$y_{TM}(T_i)$	Yield-to-maturity of the i-th security	
$y_{zc}(T_i)$	Zero-coupon yield of the i-th security	

The yield curve is composed of r par bonds with different terms. Par value is equal to $1.

The *zero-coupon yield* is the vector of *r* values $y_{zc}(T_i)$ that solves the following equations:

- If compounding is simple,Equation A, page 4-6.
- If compounding is otherwise,Equation B, page 4-6.

Equation A

$$1 + AI(T_I) = \frac{1 + c(T_i)}{\left(1 + \frac{y_{zc}(T_i)}{m}\right)^{w_i}}$$

Equation B

$$1 + AI(T_I) = \sum_{k=0}^{n_i} \frac{c(T_i)}{\left(1 + \frac{y_{zc}(L(i,\ k))}{m}\right)^{k + w_i}} + \frac{1}{\left(1 + \frac{y_{zc}(L(i,\ kn_i))}{m}\right)^{k + w_i}}$$

The yield-to maturity for the i-th security is the value yTM(Ti) that solves the following equations:

- If compounding is simple,Equation C, page 4-6.
- If compounding is otherwise,Equation D, page 4-6.

Equation C

$$1 + AI(T_I) = \frac{1 + c(T_i)}{1 + \frac{w_i}{m} y_{TM}(T_i)}$$

Equation D

$$1 + AI(T_I) = \sum_{k=0}^{n_i} \frac{c(T_i)}{\left(1 + \frac{y_{TM}(T_i)}{m}\right)^{k + w_i}} + \frac{1}{\left(1 + \frac{y_{TM}(T_i)}{m}\right)^{m_i + w_i}}$$

Related Topics

Overview of Rate Conversion, page 4-1

Conversion From Yield-to-Maturity to Zero-Coupon Yield

The system first calculates the values c(Ti) and AI(Ti) based on the following equations:

- Equation C, page 4-6
- Equation D, page 4-6

Then the system bootstraps the yield curve using the BFGS algorithm to solve the following equations:

- Equation A, page 4-6
- Equation B, page 4-6

Related Topics

Rate Conversion Algorithms, page 4-4

Conversion From Zero-Coupon Yield to Yield-to-Maturity

The system first calculates the values c(Ti) and AI(Ti) based on the following equations:

- Equation A, page 4-6
- Equation B, page 4-6

Then, the system solves these equations following the Newton-Raphson algorithm.

Related Topics

Rate Conversion Algorithms, page 4-4

5

Cash Flow Edit Logic

This chapter describes the set of logical validations that are performed, and the order in which they should be performed, on the Account table data during a Cash Flow Edits rule run.

This chapter covers the following topics:

- Overview of Cash Flow Edit Logic
- Cash Flow Edit Logic

Overview of Cash Flow Edit Logic

Oracle Transfer Pricing makes use of cash flow modeling to generate results. Inconsistent or incomplete Account table data can interrupt cash flow generation and further processing by the application.

Oracle Transfer Pricing provides a data validation and correction process, called Cash Flow Edits, to verify the accuracy and check the completeness of your Account table data and make corrections to it before you use it to generate cash flows and for further processing. See: Cash Flow Edits, *Oracle Transfer Pricing User Guide*.

The Cash Flow Edits process rule contains a set of checks to be performed on the Account table data that validate the data for consistency and ensure that the data is logically correct for handling by the cash flow engine. The set of checks and the order in which they should be performed are together known as cash flow edit logic.

Cash Flow Edit Logic

The following table details the set of checks and the order in which they are performed:

Note: The cash flow edit logic cannot be modified.

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9106	AMRT_TERM_MULT not equal to D, M, or Y	AMRT_TERM_MULT = 'M'	2	Invalid AMRT_ TERM_MULT	Checks for valid amortization term multipliers
9107	NEG_AMRT_EQ _MULT not equal to D, M, or Y	NEG_AMRT_EQ _MULT = M	2	Invalid NEG_ AMRT_EQ_ MULT	Checks for valid Neg. Amortization Eq. Multipliers
9109	ORG_TERM_MULT not equal to D, M, or Y	ORG_TERM_MULT = 'M'	2	Invalid ORG_ TERM_MULT	Checks for valid Original term multipliers
9112	PMT_FREQ_MULT not equal to D, M, or Y	PMT_FREQ_MULT = 'M'	2	Invalid PMT_ FREQ_MULT	Checks for valid payment frequency multipliers
9117	REPRICE_FREQ_ MULT not equal to D, M, or Y	REPRICE_FREQ_ MULT = 'M'	2	Invalid REPRICE_ FREQ_MULT	Checks for valid reprice frequency multipliers
9115	RATE_SET_LAG_ MULT not equal to D, M, or Y	RATE_SET_LAG_ MULT equal to 'M'	2	Invalid RATE_ SET_LAG_MULT	Checks for valid rate set lag multipliers
9111	PMT_CHG_FREQ_MULT not equal to D, M, or Y	PMT_CHG_FREQ_MULT equal to 'M'	2	Invalid PMT_ CHG_FREQ_ MULT	Checks for valid payment change frequency multipliers
9105	AMRT_TYPE_CD (<= 999 and not equal to 100, 200, 400, 500, 600, 700, 710, 800, 801, 802, 820, 830, or 999) or >29999	Set AMRT_TYPE_CD equal to '700'	2	Invalid Amortization Type	Checks that amortization type codes are either in the user defined range or that they are a valid OFSA code
9166	If REPRICE_FREQ < 0	REPRICE_FREQ = 0	2	Reprice_Freq < 0	Reprice frequencies must not be negative

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9167	If AMRT_TYPE_CD = 710 and REPRICE_FREQ <> 0	REPRICE_FREQ = 0	2	Rule of 78s Reprice_Freq<>0	Rule of 78's instruments are implicitly fixed
9155	IF INTEREST_RATE_CD < 0 or INTEREST_RATE_CD > 999	INTEREST_RATE_CD = 0	2	Int Rt Code out of range	Interest rate code must be within a valid range
9156	NET_MARGIN_CD <> 0 and NET_MARGIN_CD <> 1	NET_MARGIN_CD = 0	2	Invalid Net Margin Code	Valid net margin codes are 0 or 1
9157	T_RATE_INT_RATE_CD < 0 or T_RATE_INT_RATE_CD > 999	T_RATE_INT_RATE_CD = 999	2	T Rt Int Rt Cd out of rng	T rate interest rate code must be within a valid range
9104	ACCRUAL_BASIS_CD <1 or >6	ACCRUAL_BASIS_CD equal to '3'	2	Invalid Accrual Basis Cd	Accrual basis code must be between 1 and 6 inclusively
9114	RATE_CHG_RND_CD <0 or >4	RATE_CHG_RND_CD equal to '0'	2	Invalid Rate Chg Rnd Cd	Rate change round code must be between 0 and 4
9132	(CUR_PAYMENT < 0 and CUR_PAR_BAL > 0) or (CUR_PAYMENT > 0 and CUR_PAR_BAL < 0)	Set CUR_PAYMENT equal to 0	2	Pmt, bal opposite signs	Current payment and current balance can not have opposite signs

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9158	If CUR_BOOK_BAL <> (CUR_PAR_BAL + DEFERRED_CUR _BAL)	Warning Only	1	CurBkBl <> ParBl + Def Bl	Current book balance should equal the current
9159	CUR_NET_PAR_ BAL_C - CUR_PAR_BAL * (1 PERCENT_SOLD /100) >.001	Warning	1	NetParBl<>Com ptd NetParBl	Current net par balance should reflect the bank-owned portion of the current gross balance
9168	ORG_PAR_BAL = 0 and REPRICE_FREQ = 0	Warning	1	OrgParBal=0! Fixed Rate	For transfer pricing of fixed rate instruments, the original balance should be populated
9152	ORG_PAR_BAL < CUR_PAR_BAL and AMRT_TYPE_CD =710	message only: OrgParBal < CurParBal	1	Org Par Bal < Cur Par Bal	Original balance on rule of 78's instruments should not be greater than current balance
9160	If AMRT_TYPE_CD = 600 and PMT_DECR_CY < 0	PMT_DECR_CY = 0	2	Payment Decrease Cycle=0	Payment decrease cycle cannot be less than zero (Neg Am instruments only)
9127	ORG_PAYMENT _AMT = 0 and PMT_DECR_LIFE > 0 and AMRT_TYPE_CD = 600	Set PMT_DECR_LIFE equal to 0	2	Org Pmt=0, Pmt Dec Lf<>0	Payment decrease life is expressed as a percent of a zero original payment (Neg Am instruments only)

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9162	If AMRT_TYPE_CD = 600 and PMT_DECR_LF <> 0 and CUR_PAYMENT < ORG_PAYMENT _AMT * (1 - PMT_DECR_LF/100)	Warning Only	1	Cur Pmt < Life Pay Floor	Current payment is less than the minimum payment amount (Neg Am instruments only)
9161	If AMRT_TYPE_CD = 600 and PMT_DECR_LF < 0	Set PMT_DECR_LIFE equal to 0	2	PmtDecrLf<0 for Adj NegAm	Payment decrease life cannot be less than zero (Neg Am instruments only)
9136	AMRT_TYPE_CD = 600 and PMT_INCR_CYCLE < 0	Set PMT_INCR_CYCLE equal to 0	2	Pmt Incr Cycle < 0	Payment increase cycle cannot be less than zero (Neg Am instruments only)
9128	ORG_PAYMENT _AMT = 0 and PMT_INCR_LIFE > 0 and AMRT_TYPE_CD = 600	Set PMT_INCR_LIFE equal to 0	2	Org Pmt=0, Pmt Incr Lf<>0	Payment increase life is expressed as a percent of a zero original payment (Neg Am instruments only)
9163	If AMRT_TYPE_CD = 600 and PMT_INCR_LF < 0	Set PMT_INCR_LIFE equal to 0	2	PmtIncrLf<0 for Adj NegAm	Payment increase life cannot be less than zero (Neg Am instruments only)

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9164	If AMRT_TYPE_CD = 600 and PMT_INCR_LF <> 0 and CUR_PAYMENT > ORG_PAYMENT_AMT * (1 + PMT_INCR_LF/100)	Warning Only	1	Cur Pmt > Life Pay Cap	Current payment is greater than the maximum payment amount (Neg Am instruments only)
9141	RATE_DECR_CYCLE < 0	RATE_DECR_CYCLE equal to 0	2	Rate Decr Cycle < 0	Rate decrease cycle must not be negative
9144	RATE_FLOOR_LIFE > CUR_NET_RATE	message only: Rt Floor Lf > Cur Net Rt	1	Rt Floor Lf > Cur Net Rt	Rate floor life must not be greater than the current net rate
9145	RATE_INCR_CYCLE < 0	RATE_INCR_CYCLE equal to 0	2	Rate Incr Cycle < 0	Rate increase cycle can not be less than 0
9103	CUR_NET_RATE < 0	message only: Current Net Rate < 0	1	Current Net Rate < 0	Current net rate must not be negative
9102	CUR_GROSS_RATE < 0	message only: Current Gross Rate < 0	1	Current Gross Rate < 0	Current gross rate must not be negative
9108	(NEG_AMRT_LIMIT >= 200)or(NEG_AMRT_LIMIT < 0) and AMRT_TYPE_CD = 600	Set NEG_AMRT_LIMIT equal to 0	2	Invalid NgAm Limit	Neg Am limit value does not fall in a valid range (Neg Am instruments only)
9140	RATE_CHG_MIN < 0	Set RATE_CHG_MIN equal to 0	2	Rate Chg Min < 0	Minimum rate change can not be negative

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9139	RATE_CAP_LIFE < CUR_NET_RATE and RATE_CAP_LIFE <>0	message only: Rt Cap Life < Cur Net Rt	1	Rt Cap Life < Cur Net Rt	Current net rate is greater than the rate cap
9118	RATE_CHG_RND_FAC < 0 or RATE_CHG_RND_FAC > 1	Set RATE_CHG_RND_FAC equal to 0	2	Invalid Rt Chg Rnd Fact	Rate change round factor must be a percentage
9138	RATE_CAP_LIFE < CUR_GROSS_RATE and RATE_CAP_LIFE <> 0 and CUR_GROSS_RATE <> 0 and TEASER_END_DATE < AS_OF_DATE	Set RATE_CAP_LIFE equal to CUR_GROSS_RATE	1	Rt Cap Lf < Cur Gross Rt	Current gross rate is greater than the rate cap life
9143	RATE_FLOOR_LIFE > CUR_GROSS_RATE and CUR_GROSS_RATE <> 0 and TEASER_END_DATE < AS_OF_DATE	Set RATE_FLOOR_LIFE equal to CUR_GROSS_RATE	1	Rt Floor Lf > Cur Grss Rt	Current gross rate is less than the rate floor
9130	(ORIGINATION_DATE < 01/01/1950) or(ORIGINATION_DATE >01/01/2099)	Set ORIGINATION_DATE = 01/01/1950	2	Orig. Dt < 01/01/1950	Origination date must be acceptable
9165	If ISSUE_DATE > ORIGINATION_DATE	Set ISSUE_DATE = ORIGINATION_DATE	2	Issue Date > Orig Date	Issue date can not be greater than origination date

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9124	ORIGINATION_DATE >= AS_OF_DATE and ORIGINATION_DATE >= NEXT_PAYMENT_DATE	NEXT_PAYMENT_DATE = ORIGINATION_DATE + 1 day	2	Next Pmt Dt < Org Dt	Next payment date is less than origination date (future origination case)
9123	AS_OF_DATE > ORIGINATION_DATE and AS_OF_DATE >= NEXT_PAYMENT_DATE	NEXT_PAYMENT_DATE equal to AS_OF_DATE + 1 day	2	Next Pmt Dt < As of Dt	Next payment date is less than as-of-date (past origination case) As of Date can not be greater than the origination date and greater than the next payment date
9125	ORIGINATION_DATE <= AS_OF_DATE and NEXT_REPRICE_DATE <= AS_OF_DATE and REPRICE_FREQ > 0	NEXT_REPRICE_DATE equal to AS_OF_DATE + 1 day	2	Next Repr Dt < As of Dt	Next reprice date is less than as-of-date (past origination case)
9126	ORIGINATION_DATE > AS_OF_DATE and NEXT_REPRICE_DATE < ORIGINATION_DATE and REPRICE_FREQ > 0	Set NEXT_REPRICE_DATE equal to ORIGINATION_DATE + 1 day	2	Next Repr Dt < Org Dt	Next reprice date is less than the origination date (future origination case)

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9169	If REPRICE_FREQ <> 0 and TEASER_END_DATE > ORIGINATION_DATE and TEASER_END_DATE > AS_OF_DATE and NEXT_REPRICE_DATE > TEASER_END_DATE	Set NEXT_REPRICE_DATE equal to TEASER_END_DATE	2	Next Repr Dt > Ts End Dt	Next reprice date is greater than tease end date
9121	If AMRT_TYPE_CD = 600 and NEG_AMRT_EQ_FREQ > 0 and NEG_AMRT_EQ_DATE <= MAX(ORIGINATION_DATE, AS_OF_DATE)	Set NEG_AMRT_EQ_DATE equal to NEXT_REPRICE_DATE	2	NgAm Eq Dt < Org Dt	Neg Am equalization date is less than origination date (future origination) or less than the as-of-date (past origination) (Neg am instruments only)
9147	REMAIN_NO_PMTS_C < 1	REMAIN_NO_PMTS_C = 1	2	Rem No Pmts < 1	There has to be at least 1 payment left
9119	MATURITY_DATE < NEXT_PAYMENT_DATE	MATURITY_DATE = NEXT_PAYMENT_DATE +((REMAIN_NO_PMTS_C-1)/PMT_FREQ)	2	Mat Dt < Next Pmt Dt	Maturity date can not be before the next payment date

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9134	If AMRT_TYPE_CD = 600 and PMT_CHG_FREQ > 0 and AS_OF_DATE < PMT_ADJUST_DATE and PMT_ADJUST_DATE < ORIGINATION_DATE	PMT_ADJUST_DATE = NEXT_REPRICE_DATE	2	Pmt Adj Dt < Org Dt	Payment adjustment date is less than origination date (future origination) (Neg am instruments only)
9133	If AMRT_TYPE_CD = 600 and PMT_CHG_FREQ > 0 and AS_OF_DATE > PMT_ADJUST_DATE	PMT_ADJUST_DATE = NEXT_REPRICE_DATE	2	Pmt Adj Dt < As of Dt	Neg Am equalization date is less than the as-of-date (past origination) (Neg am instruments only)
9154	REPRICE_FREQ <> 0 and LAST_REPRICE_DATE > NEXT_REPRICE_DATE	LAST_REPRICE_DATE = NEXT_REPRICE_DATE minus REPRICE_FREQ	2	LastReprDt>NextReprDt	Last reprice date is greater than next reprice date
9148	RATE_SET_LAG < 0	RATE_SET_LAG = 0	2	Set Lag < 0	Rate set lag can not be negative
9120	NEG_AMRT_EQ_FREQ < 0 and AMRT_TYPE_CD = 600	NEG_AMRT_EQ_FREQ equal to 0	2	NegAmEqFrq < 0	Neg Am Equalization frequency cannot be negative (Neg Am instruments only)
9153	REPRICE_FREQ = 0 and (AMRT_TYPE_CD = 500 or AMRT_TYPE_CD = 600)	message only: AdjAmrtType, ReprFrq=0	1	AdjAmrtType, ReprFrq=0	Reprice frequency denotes fixed on adjustable amortization types (Conventional Adjustable and Adjustable Neg Am)

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9110	If AMRT_TYPE_CD = 600 and PMT_CHG_FREQ < 0	PMT_CHG_FREQ equal to 0	2	Invalid Pmt Chg Frq	Payment change frequency cannot be negative (Neg Am instrument only)
9131	PMT_FREQ > calculated original term	PMT_FREQ equal to calculated original term	2	Payment Freq > Org Term	Payment frequency cannot be greater than original term
9129	ORG_TERM <> calculated original term within 45 days or ORG_TERM = 0	Set ORG_TRM = calculated original term	2	Org Term<>Mat Dt -Org Dt	Original term should equal the time between the origination date and the maturity date
9135	PMT_FREQ <= 0 and ORIGINATION_DATE <= AS_OF_DATE and MATURITY_DATE > AS_OF_DATE ORIGINATION_DATE > AS_OF_DATE and MATURITY_DATE > ORIGINATION_DATE	NEXT_PAYMENT_DATE = MATURITY_DATE and ORG_TERM = calculated original term and PMT_ FREQ = calculated original term and REMAIN_NO_PMTS_C = 1	2	Pmt Freq <= 0, Trm Assumed	Payment frequency is less than or equal to zero, and both maturity date and origination date are valid dates and can be used to calculate payment frequency

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9135	PMT_FREQ <= 0 and ORIGINATION_DATE <= AS_OF_DATE, MATURITY_DATE<= AS_OF_DATE but NEXT_PAYMENT_DATE > AS_OF_DATE or ORIGINATION_DATE> AS_OF_DATE, MATURITY_DATE < ORIGINATION_DATE, but NEXT_PAYMENT_DATE > ORIGINATION_DATE	MATURITY_DATE = NEXT_PAYMENT_DATE and ORG_TERM = calculated original term and PMT_FREQ = calculated original term and REMAIN_NO_PMTS_C = 1	2	Pmt Freq <= 0, Trm Assumed	Payment frequency is less than or equal to zero and maturity date is invalid, but next payment date can be used to calculate a valid payment frequency
9135	PMT_FREQ <= 0 and ORIGINATION_DATE <= AS_OF_DATE, MATURITY_DATE<= AS_OF_DATE, and NEXT_PAYMENT_DATE <= AS_OF_DATE or ORIGINATION_DATE> AS_OF_DATE, and both MATURITY_DATE and NEXT_PAYMENT_DATE less than ORIGINATION_DATE	MATURITY DATE = AS OF DATE + 1 day and NEXT PAYMENT DATE = AS OF DATE + 1 day and ORG TERM = calculated original term and PMT_FREQ = calculated original term and REMAIN_NO_PMTS_C = 1	2	Pmt Freq <= 0, Trm Assumed	Payment frequency less than or equal to zero and all dates which can be used to calculate payment frequency are in the past

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9100	If ((AMRT_TYPE_CD <> 700 or (AMRT_TYPE_CD = 700 and AMRT_TERM <> 0)) and AMRT_TERM (in days) < ORG_TERM (in days)	Set AMRT_TERM equal to ORG_TERM, AMRT_ TERM_MULT = ORG_ TERM_MULT	2	Org Term > Amrt Term	Amortization term can only be equal to zero on Non/Amortizing instruments
9170	If REMAIN_NO_PMTS_C + (REMAIN_TERM_C (in days) / REPRICE_FREQ (in days)) >2000	Warning	1	>2000 Events for Record	The maximum number of events that can be modeled has been exceeded
9171	LRD_BALANCE= 0	LRD_BALANCE = CUR_ PAR_BAL	2	LRDBalance=0	The balance as of the last reprice date cannot be equal to 0
9172	ADJUSTABLE_TYPE_CD <> 0 and LAST_REPRICE_ DATE < ISSUE_DATE	Message Only	1	LastRepriceDate <IssueDate	When the last reprice date is less than the issue date, transfer pricing will not occur
9174	REPRICE_FREQ > 0 and ADJUSTABLE_TYPE_CD = 0	Message Only	1	AdjType=0 & ReprFreq>0	Reprice frequency and adjustable type code are inconsistent
9175	REPRICE_FREQ = 0 and ADJUSTABLE_TYPE_CD <> 0	Message Only	1	AdjType<>0 & ReprFreq=0	Reprice frequency and adjustable type code are inconsistent
9176	AMRT_TYPE_CD = 100 and ADJUSTABLE_TYPE_CD <> 0	Message Only	1	AdjType<>0 & AmrtType=100	Variable adjustable type on Conventional Fixed instrument

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9177	AMRT_TYPE_CD = 710 and ADJUSTABLE_TYPE_CD <> 0	ADJUSTABLE_TYPE_CD =0	2	AdjType<>0 & AmrtType=710	Rule of 78's instrument should only have a fixed adjustable type code
9178	(AMRT_TYPE_CD = 500 or AMRT_TYPE_CD = 600) and ADJUSTABLE_TYPE_CD = 0	Message Only	1	AdjType=0 & Amrt(500,600)	Fixed adjustable type code on adjustable amortization codes (Conventional Adjustable and Adjustable Neg Am)
9179	LAST_PAYMENT_DATE > NEXT_PAYMENT_DATE and PAYMENT_FREQ > 0	LAST_PAYMENT_DATE = NEXT_PAYMENT_DATE - PMT_FREQ	2	LastPayDate>=NextPayDate	Last payment date is greater than next payment date and can be calculated using payment frequency
9179	LAST_PAYMENT_DATE > NEXT_PAYMENT_DATE and PAYMENT_FREQ <= 0	LAST_PAYMENT_DATE = ORIGINATION_DATE	2	LastPayDate>=NextPayDate	Last payment date is greater than next payment date, but cannot be calculated using payment frequency
9180	LAST_PAYMENT_DATE < ORIGINATION_DATE	LAST_PAYMENT_DATE = ORIGINATION_DATE	2	LastPayDate<OrigDate	Last payment date can not be less than the origination date
9181	LAST_PAYMENT_DATE > AS_OF_DATE and ORIGINATION_DATE <= AS_OF_DATE	LAST_PAYMENT_DATE = AS_OF_DATE	2	LastPayDate>AsOfDate	Last payment date cannot be greater than the as-of-date if the instrument originated in the past
9182	INT_TYPE = 2 and AMRT_TYPECD = 100, 400, 500, 600 or 800	INT_TYPE = 1	2	IntType<>1,2 / Amrt Type	Interest type can only be arrears for conventionally amortizing instruments

Error Code	Error Condition	Assignment	Error Level	Warning	Description/Purpose
9182	INT_TYPE <> 1 or 2	INT_TYPE = 1	2	IntType<>1,2 / Amrt Type	Interest type must be a valid OFSA code
9183	COMPOUND_BASIS_CD <> 110, 120, 130, 140, 150, 160, 170 or 200	COMPOUND_BASIS_CD = 160	2	Invalid CompBasisCode	Compounding basis code must be a valid OFSA code
9184	(ACCRUAL_BASIS_CD = 1, 4 or 5) and (PMT_FREQ_MULT = D or AMRT_TYPE_CD = 800, 801, or 802)	ACCRUAL_BASIS_CD = 3	2	AmrtType/Accrl Basis Error	Accrual basis code cannot have a 30 day month assumption on instruments defined by a payment schedule

Related Topics

Overview of Cash Flow Edit Logic, page 5-1

6

Cash Flow Calculations

This chapter discusses cash flow calculation concepts and processes.

This chapter covers the following topics:

- Overview of Cash Flow Calculations
- Cash Flow Calculations Characteristics and Concepts
- The Cash Flow Calculation Process
- Detail Cash Flow Data
- Financial Element Calculations
- Example of the Rule of 78

Overview of Cash Flow Calculations

Oracle Transfer Pricing makes use of the OFS Cash Flow Engine to ensure modeling consistency across Oracle Financial Services (OFS) applications. An understanding of the Oracle Transfer Pricing cash flow calculations requires knowledge of cash flow calculation concepts and processes as well as that of the capabilities of the OFS Cash Flow Engine and the calculations that can be performed by it.

> **Note:** As cash flows for an instrument are modeled through time, the data associated with the instrument changes due to payments, repricings, and other characteristics or events. These changes apply only in memory and do not affect the information stored in the Account Tables.
>
> In this exposition, a subscript notation of *m* for memory and *r* for detail record helps differentiate between forecasted and actual data. For example, Current payment$_{\mathbf{r}}$ refers to the current payment stored in the Account Table. Current payment$_{\mathbf{m}}$ refers to the current payment in memory that is updated each time a payment recalculation occurs.

Cash Flow Calculations Characteristics and Concepts

A complete understanding of Oracle Transfer Pricing cash flow calculations requires familiarity with the following key concepts and characteristics:

Related Topics

Overview of Cash Flow Calculations, page 6-1

Instrument Level Modeling

Several transfer pricing methods require cash flows to generate results. Oracle Transfer Pricing supports the following cash flow transfer pricing methods:

- Duration
- Weighted Average Cash Flow
- Zero Coupon Pricing

Oracle Transfer Pricing generates cash flows at the individual instrument level. Each individual instrument record processed generates a set of cash flows as defined by that instrument record's product characteristics. The cash flows produced are then processed further to generate the required results such as transfer rates or option costs.

The usage of cash flows generated from individual instruments for processing provides an optimum level of accuracy.

Related Topics

Cash Flow Calculations Characteristics and Concepts, page 6-2

Modeling Flexibility

Oracle Transfer Pricing instrument data provides for modeling flexibility. Each instrument record has the following data:

- Payment information (dates, frequencies, and amounts)
- Balance information (current and original balance)
- Rate information (gross, net, and transfer rate)

Specific cash flow characteristics defined in an instrument record determine the cash flows to be generated. There are more than 50 data columns that directly affect the cash flow results. Depending on the information in these columns, you can model an unlimited number of instruments.

Related Topics

Cash Flow Calculations Characteristics and Concepts, page 6-2

Daily Cash Flows

Oracle Transfer Pricing lets you set instrument records in such a manner that cash flows can occur on any date, and with any frequency thereafter. This functionality is essential for generating accurate results.

Related Topics

Cash Flow Calculations Characteristics and Concepts, page 6-2

Event-Driven Logic

Oracle Transfer Pricing generates cash flows as a series of events. On any day, and with any frequency, depending on the instrument characteristics, any of the following events can occur:

- Payment
- Repricing
- Payment recalculation
- Prepayment

Each of these events requires cash flow calculations.

Related Topics

Cash Flow Calculations Characteristics and Concepts, page 6-2

Financial Elements

On an event date, such as a payment or repricing event date, the OFS Cash Flow Engine

computes the results of that event. The results of an event are represented in the form of financial elements. For example, on a payment event date, the Cash Flow Engine can compute the following:

- Interest
- Principal runoff
- Total cash flow
- Ending balance

The cash flow engine generates and outputs over 50 financial elements to the FTP_PROCESS_CASH_FLOWS table that can be used to validate results.

Related Topics

Cash Flow Calculations Characteristics and Concepts, page 6-2

The Cash Flow Calculation Process

The cash flow calculation process comprises the following broad stages:

1. Initialization of the modeling data and parameters for the instruments to be modeled. See: Initialization of Modeling Data and Parameters, page 6-4.
2. Processing of modeling events until current date equals the maturity date or the modeling end date, or the current balance equals zero. This stage involves the following steps:
   - Calculation of changes to the underlying instruments
   - Calculation of financial elements associated with an event
   - Computation of the next event date

See: Processing Modeling Events, page 6-13.

Related Topics

Overview of Cash Flow Calculations, page 6-1

Initialization of Modeling Data and Parameters

The first step in the cash flow calculation process is to gather the information necessary to model an instrument. This information is available from several sources, including:

- Account tables

- Payment Schedule table
- Payment Pattern interface
- Reprice Pattern interface
- The rules specified in the Process rule

Related Topics

The Cash Flow Calculation Process, page 6-4

Determination of Account Type

The Extended Account Type attribute of the Line Item Dimension Member determines the account type of an individual record. Account types:

- Classify instruments based on their use in financial statements.
- Determine the cash flow processing types of the instruments: detail cash flow, balance only, or interest only.

The following table illustrates the different cash flow processing types and associated account types.

Cash Flow Processing Types

Cash Flow Types	Process Description - Detail	Associated Account Type
Detail cash flow	Processes daily cash flow events and generates necessary financial elements.	Interest-Earning Asset Interest-bearing Liability Off Balance Sheet Receivable Off Balance Sheet Payable
Balance Only	Processes the record originating on the origination date and running off on the maturity date. No payments are processed.	Other Asset Other Liability

Cash Flow Types	Process Description - Detail	Associated Account Type
Interest Only	Processes the instrument as a single interest payment covering the time from the origination date to the maturity date. Recognizes the current balance as an interest cash flow on the maturity date, but accrues interest from the origination date to the maturity date.	Interest Income Interest Expense

Note: For information about financial elements output by account types, see the Financial Element Calculations table, page 6-43.

Related Topics

Initialization of Modeling Data and Parameters, page 6-4

Initialization of Interface Data

Data retrieved from the interface impacts how an instrument is processed. The interface data that affect processing are as follows:

- User Defined Payment Patterns
- User Defined Repricing Patterns

User Defined Payment Patterns

Payment pattern data is retrieved when the cash flow engine needs to process an instrument that has a payment pattern code ranging from 1000 to 29999 as its amortization type code.

The amortization type code from the detail instrument record is matched to the set of payment pattern data with the same code. If no match is found for the amortization code from the detail record, the instrument record is assumed to be of nonamortizing type.

User Defined Repricing Pattern

Repricing pattern data is retrieved when the cash flow engine needs to process an instrument that has a repricing pattern code ranging from 500 to 998 as its adjustable type code.

If no related repricing pattern data is found, the engine defaults to the record characteristics in the repricing frequency column and processes the instrument as a standard adjustable instrument.

When a match is made, the instrument is modeled based on the repricing pattern. The cash flow engine first evaluates the status of the instrument as of the starting date of the cash flow process. The current repricing date is determined by rolling forward from the origination date to the next repricing date that follows the process start date. If the process start date does not correspond to the next repricing date, the repricing date from the record is used.

A repricing event is triggered when the period between two repricing events has elapsed. When this occurs, the defined rates are assigned to the detail record of the instrument. If the repricing type is Flat Rate, the rate from the event detail of the repricing pattern is applied to the detail record of the instrument. If the repricing type is Indexed Rate, a rate lookup is triggered for the customer rate and the transfer pricing rate. If the interest rate code (IRC) is a yield curve, the point on the yield curve used is the repricing term associated with the current repricing information, unless the IRC term has been specified in the repricing pattern event. This rate, plus the specified margin, is the new fully indexed rate. Rate caps and floors are applied after this calculation occurs.

Related Topics

Initialization of Modeling Data and Parameters, page 6-4

Initialization of Cash Flow Data

The cash flow engine gathers the cash flow data for the instrument to be processed. The cash flow data for an instrument is a subset of the information stored in the Account Tables for that instrument record. For a complete list of columns referenced in the cash flow calculation process, see theFinancial Element Calculations table, page 6-35.

The cash flow data provides current information about the characteristics of a cash flow instrument. This information must be consistent to ensure accurate output. Before processing cash flows, the cash flow edits checks should be run to avoid producing unreasonable results. See:

- Cash Flow Edits, *Oracle Transfer Pricing User Guide*.
- Overview of Cash Flow Edit Logic, page 5-1.

Cash flow data can be classified into the following three categories defining its use during the processing of an event:

- Static Characteristics
- Dynamic Characteristics
- Triggers

Static Characteristics

Static characteristics provide information to the cash flow engine about how the

instrument should be modeled. For non-pattern and non-schedule instruments, all of the following characteristics remain constant during the modeling process:

- **Event Frequencies:**
  - Repricing
  - Payment
  - Payment Change
- **Financial Code Values:**
  - Accrual Basis Code
  - Amortization Type Code
  - Rate Change Round Code
- **Dimensions:**
  - Line Item Dimensions Members
  - Line Item Hierarchy
  - Currency Codes
- **Repricing Parameters:**
  - Margin
  - Rate Caps
  - Rate Increase Period
  - Rate Change Minimum

For pattern and schedule instruments, the payment frequency can vary throughout the life of the instrument.

Dynamic Characteristics

Dynamic characteristics are updated each time an event occurs, as a result of what has occurred during the event. These include the following:

- **Balances:**
  - Current
  - Current Deferred

- **Rates:**
  - Current Net
  - Current Gross
  - Current Transfer
- **Event Counters (remaining number of payments)**

Triggers

Triggers signal the cash flow engine when it is time to model a particular event and can change their value during the modeling process. These include the following:

- **Event Dates:**
  - Next Payment
  - Next Reprice
  - Payment Change
- **Negative Amortization (NGAM or neg-am) Balance (in conjunction with neg-am limit)**

Related Topics

Initialization of Modeling Data and Parameters, page 6-4

Initialization of Schedule Records

Oracle Transfer Pricing provides a payment schedule table, FTP_PAYMENT_SCHEDULE, for loading cash flow details related to instruments that have unique, non-standard payment characteristics which cannot easily be captured using traditional amortization methods or into a payment pattern definition. To properly model scheduled-amortization instruments, the cash flow engine must retrieve payment dates and payment amounts from the payment schedule table.

The payment schedule table is referenced by the Cash Flow engine by lookup, based on a match of the Instrument type code (INSTRUMENT_TYPE_CODE) and Account number (ID_NUMBER) columns from the instrument record to the same data in the payment schedule table. The Cash Flow engine looks at this table when the AMRT_TYPE_CODE column on the instrument record is populated with a value of 800, 801, or 802. If no match is found, the instrument is processed as a non-amortizing record.

Amortization Code 800

An amortization code of 800 denotes conventional payment schedule and signifies that

payment amounts are principal and interest amounts. On payment dates, these payments are processed as conventional amortization payments.

Amortization Code 801

An amortization code of 801 denotes level principal payment schedules. On the schedules for these instruments, the payment amount comprises the principal portion of the payment only. For both conventional (800) and level principal (801) payment schedules, the payment amounts should be expressed gross of any participations.

Amortization Code 802

An amortization code of 802 denotes simple interest payment schedule. Instruments with amortization codes of 802 do not reference the principal amount column in the schedule tables. These instruments are processed as interest only records, with all principal, with the exception of prepayments, running off on the maturity date.

The data in the maturity date, next payment date, last payment date, remaining number of payments, and current payment columns of the instrument record should coincide with the same information in the schedule table. When this information is inconsistent, the information in the detail record supersedes the data in the schedule table.

In this case, the payment on the next payment date occurs on the date defined in the next payment date column of the instrument record, for the amount defined in the current payment column of the instrument record.

All payments after this date and before the maturity date are made according to the payment date in the schedule table. On the maturity date, the date from the maturity date column of the instrument record is used to pay off the remaining balance of the instrument record. If payment dates exist in the schedule beyond this date, they are ignored.

Related Topics

Initialization of Modeling Data and Parameters, page 6-4

Initialization of Pattern Records

The following logic applies to both user defined payment and repricing patterns.

Single Timeline Patterns

To initialize an instrument record whose payment (or repricing) characteristics are defined by a single timeline pattern, the cash flow engine must synchronize the detail instrument record with the payment (or repricing) pattern. Synchronization determines the current payment of the instrument within the payment (or repricing) pattern.

The synchronization process depends on whether the pattern is relative or absolute.

To synchronize a relative pattern, the cash flow engine calculates the payment (or repricing) dates for the instrument record by rolling the origination date forward by the pattern frequencies. Once it calculates a payment (or repricing) date greater than the

as-of-date, it stops. The number of times it was necessary to roll the date forward determines the current payment (or repricing event) number for the record.

Example

An instrument record processed on an as-of-date of 03/31/1996 with an origination date of 01/01/1996, and a next payment (or repricing) date of 05/15/1996 is matched to the pattern described in the following table:

Example of Single Timeline Pattern

Row #	Frequency	Repetitions
1	1 M	3
2	3 M	3

The origination date is rolled forward in the following manner from the Starting point - 01/01/1996:

- Add first monthly payment (or repricing) frequency - 02/01/1996
- Add second monthly payment (or repricing) frequency - 03/01/1996
- Add third monthly payment (or repricing) frequency - 04/01/1996

After the third roll forward, the payment (or repricing) date is greater than the as-of-date. The cash flow engine interprets that the record is on its third payment (or repricing), which is the final monthly payment (or repricing). It models this payment (or repricing) on the next payment (or repricing) date from the detail record, in this case, 05/15/1996. The next payment (or repricing) is scheduled for 8/15/1996, using the three month frequency from the fourth payment (or repricing) in the schedule.

Absolute patterns do not require the same rolling mechanism for synchronization. The next payment (or repricing) date from an absolute pattern is determined by the first month and day after the as-of-date. If this date does not correspond to the next payment (or repricing) date from the detail record, the next payment (or repricing) date of the detail record supersedes the date of the pattern. From that point on in the process, the payment (or repricing) dates from the pattern are used.

The cash flow engine has been designed in this manner to allow greater flexibility in modeling payment and repricing patterns. However, this flexibility increases the importance of detail data accuracy to ensure that when discrepancies exist between detail data and patterns, the differences are intended.

Multiple Timeline Patterns (Payment Patterns Only)

To initialize a detail instrument record tied to a split pattern, the cash flow engine generates a separate record for each split. The current balance for each split record is

calculated using the percentage apportioned to that split, as defined through the payment pattern interface. The original balance, original payment, and current payment columns are also apportioned according to the percentage defined through the interface.

For each timeline resulting from the split of a detail instrument record, the current payment date must be determined. The method for determining the payment date is the same as described for single timeline patterns with one exception. For these instruments, the next payment date from the original instrument record does not override the calculated next payment date. The date derived from rolling the origination date forward for relative timelines or locating the next date for absolute timeliness is assumed to be the correct payment date.

Related Topics

Initialization of Modeling Data and Parameters, page 6-4

Determination of Modeling Start and End Dates

Modeling start and end dates are determined by the type of processing and the instrument being processed, as shown in the following table:

Modeling Start and End Dates

Processing and instrument type	Start Sate	End Date
OFS Transfer Pricing - Adjustable	last reprice date	next reprice date
OFS Transfer Pricing - Fixed	origination date	maturity date
OFS Transfer Pricing- Adjustable Remaining Term	effective date + 1 day	next reprice date
OFS Transfer Pricing- Adjustable Remaining Term	effective date + 1 day	maturity date

Only records that have a value in the Calendar Period column of the database equal to the Calendar Period input as a run parameter for the Transfer Pricing Process rule are included.

Related Topics

Initialization of Modeling Data and Parameters, page 6-4

Initialization of Additionally Derived Data

The transfer pricing of adjustable rate instruments involves the following additional

steps:

- Data reset to values consistent with the last reprice date
- Percent sold adjustment

Data Reset to Values Consistent with the Last Reprice Date

Oracle Transfer Pricing resets data to values consistent with the last reprice date by rolling back the next payment date by the payment frequency to the first payment date after the last reprice date. The application increases the remaining number of payments by the number of payments added in the rollback process.

The Last Reprice Date Balance is used in place of the current balance in a transfer pricing process. If the balance as of the last reprice date is not available, the application updates this column with the current balance. Oracle Transfer Pricing program has a special feature that re-amortizes the current payment if the following conditions are met:

- The last reprice date balance equals the current balance
- Payments occur between the last reprice date and the Calendar Period End Date
- The instrument is not tied to an amortization pattern or an amortization schedule

These three conditions signal the cash flow engine that the balance as of the last reprice date is not available and that the current balance should be used as a proxy.

Percent Sold Adjustment

Balances must be adjusted for participations:

- Current net balance = current par balance * (100 - percent sold)
- Current payment net = current payment * (100 - percent sold)
- Original net balance = original par balance * (100 - percent sold)
- Last reprice balance net = last reprice balance * (100 - percent sold)
- Original payment net = original payment * (100- percent sold)

Related Topics

Initialization of Modeling Data and Parameters, page 6-4

Processing Modeling Events

The cash flow engine models the following four events:

- Payment
- Payment Change
- Reprice
- Prepayment

When multiple events occur on the same day the cash flow engine processes them in the following order:

- Interest in Arrears
  1. Payment Calculation
  2. Payment
  3. Prepayment
  4. Reprice
- Interest in Advance
  1. Reprice
  2. Payment
  3. Prepayment

  Note: For interest in advance instruments, payment calculation is not applicable. Payment calculation occurs only on conventionally amortizing instruments.

The cash flow engine processes an event as follows:

- Updates the dynamic information.
- Generates financial elements summarizing the event.
- Increments event dates to the next event date.

Related Topics

The Cash Flow Calculation Process, page 6-4

Payment Calculation Event

The cash flow data for payment calculation has the following characteristics:

Static Information: Conventional Adjustable and Payment Patterns

- Current Gross Rate
- Current Par Balance
- Amortization Term And Multiplier
- Amortization Type
- Adjustable Type Code
- Compounding Basis Code

Additional Information: Adjustable Negative Amortization (NGAM):

- Payment Increase Cycle
- Payment Increase Life
- Payment Decrease Cycle
- Payment Decrease Life
- Payment Change Frequency And Multiplier
- NGAM Equalization Frequency And Multiplier
- Original Payment Amount
- Dynamic Information
- Current Payment

Event Trigger: Transfer Pricing

- Cash flow transfer pricing of an Adjustable instrument

Event Triggers: Conventional Adjustable and Conventional Payment Patterns

- Reprice Event

Event Triggers: Adjustable Negative Amortization (NGAM)

- Next Payment Change Date
- NGAM Balance > NGAM Limit

- NGAM Equalization Date

Related Topics

Processing Modeling Events, page 6-13

Payment Calculation Steps

Payment calculation comprises the following steps:

1. Calculation of New Current Payment, page 6-16
2. Application of Periodic Payment Change Limits, page 6-17
3. Application of Lifetime Payment Change Limits, page 6-18
4. Negative Amortization (NGAM) Equalization, page 6-18
5. Update of Current Payment Field, page 6-18

1. Calculation of New Current Payment

Conventionally Amortizing Payment = Current Par $Balance_m$ / (1 / Current $Rate_c$) x (1 - (1 + Current $Rate_c$) ^ rem $pmts_a$)

Here:

- Current $Rate_C$ = current compounded customer rate per payment
- rem $pmts_a$ = remaining number of payments based on amortization
- Current Par $Balance_m$ = current balance at time of payment recalculation

For conventional schedules that reprice, payment recalculation does not occur. For patterns which reprice, payment recalculation does not occur during the repricing event. For these instruments, the payment is calculated at the time of payment. See: Payment Event, page 6-18.

Current Compounded Customer Rate per Payment

The customer rate must be adjusted to a rate per payment. If no compounding occurs, the rate can be divided by the payments per year. The following table illustrates how a current customer rate of 7.5% is adjusted to a rate per payment.

Example of Current Customer Rate Adjusted to a Rate per Payment

Payment Frequency	Calculation	Rate per Payment
monthly	7.5 ÷12	0.625

Payment Frequency	Calculation	Rate per Payment
quarterly	7.5 ÷ 4	1.875
yearly	7.5 ÷ 1	7.5

If the instrument compounds, the rate must be adjusted for compounding. For monthly rates that compound daily, an average number of days assumption of 30.412 is used.

Remaining Number of Payments Based on Amortization

If the amortization term is equal to the original term, the calculation of the remaining number of payments is simple.

However to calculate the remaining number of payments when the amortization term is lesser or greater than the original term, the cash flow engine adds the amortization term to the origination date to determine the amortization end date. The cash flow engine calculates the remaining number of payments by determining the number of payments that can be made between the next payment date and the amortization end date, and adding one additional payment for the payment on the next payment date.

The cash flow engine calculates the remaining number of payments for patterns based on the payment frequency at the time of repricing. As with conventional instruments, the amortization end date is used for payment recalculation. The remaining term is calculated using the difference between the next payment date and the amortization end date. This term is divided by the active payment frequency and one additional payment is added to it for the payment on the next payment date.

2. Application of Periodic Payment Change Limits

Periodic payment change limits restrict the amount by which a payment installment can increase over its previous value. These limits are applied only when the payment installment recalculation is triggered by a payment adjustment date or a negative amortization limit. Because of these limits, the principal may continue to negatively amortize when the negative amortization limit has been reached. The following table describes payment conditions and related adjustments.

Payment Conditions and Related Adjustments

Condition	Adjustment
Increasing Payment: Newly Calculated Payment > (1 + (Payment Increase $Life_r$/100)) * Original $Payment_r$	Current $Payment_m$ = (1+ (Payment Increase $Life_r$/100)) * Original $Payment_r$

Condition	Adjustment
Decreasing Payment: Newly Calculated Payment < (1 + (Payment Decrease $Life_r$/100)) * Original $Payment_r$	Current $Payment_m$ = (1 + (Payment Decrease $Life_r$/100)) * Original $Payment_r$

3. Application of Lifetime Payment Change Limits

Lifetime payment cap and floor set the maximum and minimum amount for a payment installment. These limits are applied only when a payment recalculation is triggered by a payment adjustment date or a negative amortization limit. Because of these limits, principal may continue to negatively amortize when the negative amortization limit has been reached.

Payment Conditions and Related Adjustments

Condition	Adjustment
Increasing Payment: Newly Calculated Payment > (1 + (Payment Increase $Life_r$/100)) * Original $Payment_r$	Current $Payment_m$ = (1+ (Payment Increase $Life_r$/100)) * Original $Payment_r$
Decreasing Payment: Newly Calculated Payment < (1 + (Payment Decrease $Life_r$/100)) * Original $Payment_r$	Current $Payment_m$ = (1 + (Payment Decrease $Life_r$/100)) * Original $Payment_r$

4. Negative Amortization (NGAM) Equalization

If a payment recalculation is triggered by a NGAM equalization date, payment change limits do not apply. If the newly calculated payment is greater than the lifetime payment cap or less than the lifetime payment floor, the appropriate lifetime payment limit (cap/floor) is set equal to the newly calculated payment.

5. Update of Current Payment Field

Once all the payment limits have been applied, the new current payment is updated in memory for the processing of future events.

Related Topics

Processing Modeling Events, page 6-13

Payment Event

The cash flow data for payment event has the following characteristics:

- Static Information
- Dynamic Information
- Event Triggers

Static Information

- Amortization Type
- Current Payment
- Accrual Basis Code
- Current Gross Rate
- Current Net Rate
- Origination Date
- Payment Frequency And Multiplier
- Interest Type
- Compounding Basis Code
- Last Payment Date

Dynamic Information

- Current Par Balance
- Remaining Number Of Payments
- Negative Amortization (NGAM) Balance

Event Triggers

- Next Payment Date
- Maturity Date

Related Topics

Processing Modeling Events, page 6-13

Payment Event Steps

The payment event comprises the following steps:

1. Calculation of Interest Cash Flows, page 6-20
2. Calculation of Current Payment for Patterns and Schedules, page 6-22
3. Calculation of Principal Runoff, page 6-24
4. Performing Special Negative Amortization Check, page 6-24
5. Identifying Maturity Date Case, page 6-25
6. Update of Current Balance, page 6-25
7. Update of Remaining Number of Payments, page 6-25
8. Update of Next Payment Date, page 6-25

1. Calculation of Interest Cash Flows

The cash flow engine calculates the amount of interest to be paid on a payment date as described in the following table:

Interest Rate Calculation Formulas

Interest Cash Flow	Calculation
Interest cash flow gross	Current net par balance * gross rate per payment
Interest cash flow net	Current net par balance * net rate per payment

> **Note:** Loans with amortization schedules based on the Rule of 78 are exceptions. These loans have a precomputed interest schedule. See: Example of the Rule of 78, page 6-48.

Rate per Payment: Accrual Adjustment

The annual coupon rate must be adjusted to a rate per payment. The accrual basis codes define how this adjustment should be made.

Accrual Basis Codes Example

The following table shows how using different accrual basis codes an annual coupon rate of 6 percent is adjusted to a rate per payment for a June 30 payment for an instrument that has a payment frequency of three months.

Accrual Basis Codes and Rate per Payment Adjustments

Accrual Basis Code	Payment Adjustment	Rate per Payment
30/360	(3*30)/360 * 6.0	1.500%
30/365	(3*30)/365 * 6.0	1.4795%
30/Actual	90/365 * 6.0	1.4795%
Actual/Actual	(30+31+30)/365 * 6.0	1.4959%
Actual/365	(30+31+30)/365 * 6.0	1.4959%
Actual/360	(30+31+30)/360 * 6.0	1.5167%

This formula assumes a single rate per payment period. If an instrument reprices multiple times within a payment period, only the last repricing event affects the interest cash flow.

Rate per Payment: Compounding Adjustment

The cash flow engine compounds the rate used in interest calculation if the compounding frequency is less than the payment frequency. The following table describes the interest rate compounding codes and the associated compounding formula.

Compounding Codes and Formula

Compounding Code	Calculation
Simple	No compounding
At Maturity	No compounding
Continuous	e^(rate per payment) -1
Daily	(1 + rate/365 or 366) ^ (days in payment) -1
Monthly, Quarterly, Semi-Annually, Annually	(1 + rate per pmt/cmp pds per pmt) ^ (cmp pds per pmt)

Rate per Payment: Stub and Extended Payment Adjustment

An adjustment may be made if the expected payment frequency (in days) is different from the actual payment frequency (in days). The number of days between the next payment date and the last payment date is compared to the payment frequency specified in days. The payment frequency specified in days depends on the month the payment occurs. If these numbers are not equal, the interest cash flow is adjusted by the following ratio:

(next payment date - last payment date)/payment frequency in days

This adjustment holds true for all payments other than the final payment. For the final payment, the number of days between the maturity date and the last payment before the maturity date is compared to the payment frequency specified in days. The payment frequency specified in days depends on the month in which the maturity date occurs. If these numbers are not equal, the interest cash flow is adjusted by the following ratio:

(maturity date - last payment date)/payment frequency in days

2. Calculation of Current Payment for Patterns and Schedules

For amortization patterns, the payment amount can be defined independently for each payment date. Therefore, on each payment, the payment amount must be calculated according to the characteristics defined for that date. The payment amounts depends on the following factors:

- Percent of Original Payment
- Percent of Current Payment
- Percent of Current Balance
- Percent of Original Balance
- Absolute Value
- Interest Only

Percent of Original Payment

You can choose to transfer price your product portfolio either in the Standard or in Remaining Term calculation mode. See: Transfer Pricing Process Rule and Calculation Modes, *Oracle Transfer Pricing User Guide*.

In the Standard calculation mode, the cash flow engine calculates the payment amount for each payment that falls between the last reprice date and the next reprice date (adjustable rate instruments) or between the origination date and maturity date (fixed rate instruments) by using the original payment from the instrument record and applying the pattern percent from the interface. The pattern percent is the percent of original payment specified in the interface for that payment.

In the Remaining Term calculation mode, the cash flow engine calculates the payment amount as follows:

On the first modeled payment on a detail instrument record, the amount in the current payment column is assumed to accurately represent the payment amount as of the next payment date. If this instrument record is partially sold, the current payment is multiplied by (100 - percent sold) to get the net payment amount.

If this is the first payment made by a new business record, the payment amount is calculated using the original balance, original rate, and original number of payments. The original number of payments is calculated by using the amortization term, as specified through the Maturity Strategy rule or Transaction Strategy rule, and the original payment frequency.

On subsequent payment dates, the cash flow engine calculates the amount paid by multiplying the pattern percent by the amount in the original payment amount column, adjusted for percent sold, if applicable.

Percent of Current Payment

On the payment date, the cash flow engine determines the amount to be paid by first calculating a new payment according to the "active" characteristics, including the current balance, current rate, current payment frequency, and calculated remaining number of payments. The remaining number of payments is calculated by determining the amount of time remaining in the amortization term and dividing this term by the current payment frequency.

After the payment has been calculated, the pattern percent is applied.

Percent of Current Balance

Percent of Current Balance is applicable only for Level Principal payment patterns. On the first modeled payment, the amount in the current payment column is assumed to accurately represent the payment amount as of the next payment date. If the instrument is partially sold, the amount should be multiplied by (100 - percent sold) to get the net payment amount. For all subsequent payments, the payment amount should be calculated at the time of payment by multiplying the outstanding balance by the pattern percent. However, for fixed rate instruments, modeling begins at the origination date, using the original balance. For adjustable rate instruments, modeling begins at the last reprice date, using the last reprice date balance.

Percent of Original Balance

Percent of Original Balance is applicable only for Level Principal payment patterns. On the first modeled payment, the amount in the current payment column is assumed to accurately represent the payment amount as of the next payment date. If the instrument is partially sold, the amount should be multiplied by (100 - percent sold) to get the net payment amount.

For all subsequent payments, the payment amount should be calculated at the time of payment by multiplying the original balance, net of participations, by the pattern percent.

Absolute Value

On the first modeled payment, the amount in the current payment column is assumed to accurately represent the payment amount as of the next payment date. If the instrument is partially sold, the amount should be multiplied by (100 - percent sold) to

get the net payment amount.

For all subsequent payments, the absolute value amount from the pattern is used. If the instrument has a percent sold, the percent sold is applied to the absolute payment amount.

For standard transfer pricing, the absolute payment amount is used, adjusted for the participation percent.

Interest Only

On all interest only payments, the payment amount is calculated as the interest due on that date. No reference is made to the current payment column from the detail instrument record. Any payments in the current payment column are ignored.

3. Calculation of Principal Runoff

Principal runoff is a function of the amortization type of the instrument and the current payment. The current payment on a conventionally amortizing record represents the total principal and interest payment, while the current payment on a level principal record represents the principal portion of the total payment.

Simple Amortization

(code = 700, 802, and any Non-Amortizing Pattern Codes)

General case: Principal Runoff = 0

Interest Credited: -1 * interest cash flow gross

Conventional Amortization

(code = 100, 500, 600, 800, and any conventionally amortizing pattern codes)

Principal Runoff = current $payment_m$ - interest cash flow gross

Level Principal Amortization

(code = 820, 801, and any level principal amortizing pattern codes)

Principal Runoff = current $payment_m$

Rule of 78 (code = 710)

Principal Runoff = current $payment_m$ - interest cash flow gross

4. Performing Special Negative Amortization Check

If principal runoff is negative and the instrument record is adjustable negative amortization, then it must be ensured that the record is not exceeding negative amortization limits. As part of special negative amortization check, the cash flow engine verifies if the following condition holds true.

-1 * principal runoff + neg am $balance_m$ > neg am $limit_t$ /100 * original $balance_r$

If this condition is true, the payment is not made. The payment is recalculated. See: Payment Event, page 6-18.

After the new payment has been calculated, the scheduled principal runoff is recalculated, based on the new payment information.

5. Identifying Maturity Date Case

If the payment date is also the maturity date, then the remaining balance must be paid off.

Principal At Maturity = Current Balance$_m$ - Scheduled Principal Runoff

6. Update of Current Balance

The current balance must be updated to reflect the principal portion of the payments and any interest credited.

Current Balance$_m$ = Current Balance$_m$ - Principal Runoff - Principal At Maturity + Interest Credited

7. Update of Remaining Number of Payments

After a payment has been made, the underlying information must be updated in preparation for the next event.

The remaining number of payments is reduced by 1. If remaining number of payments is zero, the modeling for this instrument is complete.

Remaining Payments$_m$ = Remaining Payments$_m$ - 1

8. Update of Next Payment Date

For standard amortization instruments, the next payment date is set equal to the current payment date plus the payment frequency.

Next payment date$_m$ = Current payment date + payment frequency

If an instrument has an amortization schedule, the next payment date is determined from the dates in the schedule table.

If an instrument has an amortization pattern, the next payment date is determined by incrementing the current payment date by the current payment frequency for relative patterns. For absolute patterns, the next payment date is determined by the next consecutive date in the pattern.

If the remaining number of payments is equal to 1, or the next payment date is greater than the maturity date, the next payment date is set equal to the maturity date.

Related Topics

Processing Modeling Events, page 6-13

Payment Event Steps for Interest-in-Advance Instruments

The following steps are applicable only to payment calculation for interest-in-advance records. Interest-in-advance instruments require their first payment to be made on the origination date. The last payment, made on the maturity date, is a principal only payment.

1. **Determination of New Current Payment on Schedules and Patterns**

   See: Calculation of Current Payment for Patterns and Schedules, page 6-22.

2. **Calculation of Principal Runoff**

   For interest-in-advance records, the principal runoff occurs *before* the interest cash flow is calculated. Because conventionally amortizing instruments cannot have interest-in-advance characteristics, amortizing interest-in-advance instruments are always level principal. Therefore, the principal runoff equals the current payment amount.

   For the payment on the maturity date, all remaining principal is also paid off.

3. **Update Of Current Balance**

   Before calculating the interest cash flow, the current balance must be updated for the amount of principal runoff. If the payment is the maturity date, the balance is set to zero, and no further calculations are necessary.

4. **Calculation of Interest Cash Flow**

   If the payment date is not the maturity date, an interest cash flow is made. The interest cash flow calculation for interest-in-advance instruments is similar to the interest in arrears calculation. The calculation differs in the count for number of days. Rather than counting from the last payment date to the current payment date, the number of days is counted from the current payment date to the next payment date.

5. **Update of Remaining Number of Payments**

   After a payment has been made, the underlying data must be updated in preparation for the next event. The remaining number of payments is reduced by 1.

6. **Update Next Payment Date**

   For standard amortization instruments, the next payment date is set equal to the current payment date plus the payment frequency.

   Next payment $date_m$ = Current payment date + payment frequency

   If the instrument has an amortization schedule, the next payment date is determined from the dates in the schedule table.

   If the instrument has an amortization pattern, the next payment date is determined by incrementing the current payment date by the current payment frequency for relative patterns. For absolute patterns, the next payment date is determined by the next consecutive date in the pattern.

   If the remaining number of payments is equal to 1, or the next payment date is greater than the maturity date, the next payment date is set equal to the maturity date.

Related Topics

Processing Modeling Events, page 6-13

Prepayment Event

The cash flow data for prepayment event has the following characteristics:

- Static Information
- Dynamic Information
- Additional Assumption Information
- Event Triggers

Static Information

- Current Gross Rate
- Current Net Rate
- Payment Frequency and Multiplier
- Origination Date
- Original Term
- Next Reprice Date
- Reprice Frequency
- Maturity Date

Dynamic Information

- Current Par Balance
- Current Payment

Additional Assumption Information

- Prepayment Assumption Rule
- Prepayment Table Rule
- Forecast Rates Rule

Event Triggers

- Next Payment Date

Related Topics

Processing Modeling Events, page 6-13

Prepayment Event Steps

The prepayment event comprises the following steps:

1. Update of the Value of the Prepayment Dimensions, page 6-28
2. Determination of the Base Annual Prepayment Rate, page 6-31
3. Adjustment for Seasonality, page 6-33
4. Selection of the Prepay in Full Option, page 6-33
5. De-annualization of the Prepayment Option, page 6-33
6. Adjustment of the Prepayment Rate for Stub or Extended Payments, page 6-33
7. Determination of the Prepayment Amount, page 6-33
8. Update of the Current Balance, page 6-33
9. Application of the Prepayment Factor to Current Payment, page 6-33

1. Update of the Value of the Prepayment Dimensions

Depending on the prepayment assumptions for a product, values for the prepayment dimensions may need to be updated. The prepayment assumptions for the product are defined in a Prepayment rule, which is then selected for the current processing run.

If the prepayment method is Constant Rate, these updates are not necessary. If the prepayment method is Arctangent, only the rate ratio need be calculated. For Prepayment Table method, the required updates depend on the dimension within the table for the proper origination date range.

The possible prepayment dimensions are as follows:

- Market Rate
- Coupon Rate
- Rate Difference
- Rate Ratio

- Original Term
- Reprice Frequency
- Remaining Term
- Expired Term
- Term to Reprice

Market Rate

The market rate is selected per product within the Prepayment rule. You must choose an interest rate code (IRC) from the list of IRCs contained in the active Historical Rates database. The chosen IRC provides the base value for the market rate.

Additionally, you must specify the term point you want to use for IRCs which are yield curves. There are three possible methods for you to select:

- Original Term

  The cash flow engine retrieves the rate from the term point equaling the original term on the instrument.

- Reprice Frequency

  The cash flow engine retrieves the rate from the term point equaling the reprice frequency of the instrument. If the instrument is fixed rate and, therefore, does not have a reprice frequency, the calculation retrieves the rate associated with the term point equaling the original term on the instrument.

- Remaining Term

  The cash flow engine retrieves the rate from the term point equaling the remaining term of the instrument.

  The market rate is determined by retrieving the proper rate and adding the user-input spread.

  Market Rate = f(Current Date, IRC, yield curve term) + spread

Coupon Rate

The coupon rate is the current gross rate of the instrument record (as of the current date in the forecast).

Rate Difference

The rate difference is the spread between the coupon rate and the market rate. Before calculating this dimension, the market rate must be retrieved.

Rate Difference = Coupon Rate - Market Rate

Rate Ratio

The rate ratio is the proportional difference between the coupon rate and the market rate. Before calculating this dimension, the market rate must be retrieved.

Rate Ratio = Coupon Rate/Market Rate

Original Term

The original term is retrieved from the original term of the instrument. If the original term is expressed in months, no conversion is necessary. Otherwise, the following calculations are applied:

$\text{Original Term}_{\text{months}} = \text{ROUND}((\text{Original Term}_{\text{days}})/30.412)$

$\text{Original Term}_{\text{months}} = \text{Original Term}_{\text{years}}*12$

Reprice Frequency

The value for reprice frequency depends on the adjustable type code and the tease characteristics of the instrument data.

- **Fixed Rate**

  If the instrument is fixed rate, as designated by an adjustable type code = fixed (code value = "0"), the original term, as defined above, is used as the reprice frequency.

  Reprice Frequency = Original Term (months)

- **Non-Tease Floating**

  If the adjustable type of the instrument is floating (code value of "30" or "50" and not in a tease period), the reprice frequency is assumed to be one day, which when rounded to a month value, becomes 0 months.

  Reprice Frequency = 0 months

- **Non-Tease Adjustable**

  If the adjustable type of the instrument is adjustable (code value of "250") and not in a tease period, the reprice frequency columns is used. All cases where terms are not expressed in months should be translated into months, calculated as follows:

  $\text{Reprice Frequency}_{\text{months}} = \text{Reprice Frequency}_{\text{years}}*12$

  $\text{Reprice Frequency}_{\text{months}} = \text{Round (Reprice Frequency }_{\text{days}}/30.412$

- **Teased Loans**

  The tease period is identified by a tease end date > current date. The reprice frequency during the tease period is calculated as follows (rounded to the nearest whole number of months).

  Reprice Frequency = ROUND((Tease End Date - Origination Date) /30.412)

Remaining Term

The remaining term value represents the remaining number of months until maturity. The value is rounded to the nearest whole number of months.

Remaining Term = ROUND((Maturity Date - Current Date)/30.412)

Expired Term (Age)

The expired term represents the age of the instrument. It represents the time elapsed since the origination of the instrument. The value is rounded to the nearest whole number of months.

Expired Term = ROUND((Current Date - Origination Date)/30.412)

Term to Reprice

As with reprice frequency, the calculation of term to reprice depends on the adjustable type code and tease characteristics of the instrument characteristics.

- **Fixed Rate**

  If the instrument is fixed rate, as designated by an adjustable type code = fixed (code value = "0"), the term to reprice is calculated in the same manner as remaining term. The value is rounded to the nearest whole number of months.

  Term to Reprice = ROUND (Maturity Date - Current Date/30.412)

- **Non-Tease Floating**

  If the adjustable type of the instrument is floating (code value of "30" or "50"), and is not in its tease period, the reprice frequency is taken as 1 day. The term to reprice is assumed to be one day, which when rounded to a month value, becomes 0 months.

  Term to Reprice = 0 months

- **Non-Tease Adjustable**

  If the adjustable type of the instrument is adjustable (code value of "250") and not in its tease period, the term to reprice is calculated as the difference between the current date and the next reprice date. The value is rounded to the nearest whole number of months.

  Term to Reprice = ROUND((Maturity Date - Current Date)/30.412)

- **Teased Loans**

  The tease period is identified by a tease end date > current date. The term to reprice, while in this period, is calculated as the difference between the current date and the tease end date. The value is rounded to the nearest whole number of months.

  Term to Reprice = ROUND((Tease End Date - Current Date)/30.412)

2. Determination of the Base Annual Prepayment Rate

The method for determining the annual prepayment rate depends on the prepayment method.

Constant Rate

Constant prepayment rates can vary for different origination date ranges. The rate is determined by finding the proper range of origination dates and using the constant rate from this range.

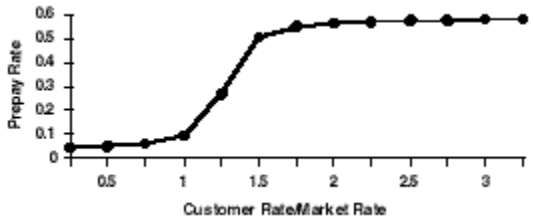
Base Annual Prepayment Rate = Constant Rate

Arctangent

The arctangent formula describes the relationship between prepayments and the ratio of coupon rate to market rate. Four coefficients, to which you enter coefficients, define the shape of the curve. These coefficients can vary by origination date range.

Base Annual Prepayment (PP) Rate = Coeff1 - Coeff2 * ARCTANGENT(Coeff3* (Coeff4 - Rate Ratio))

The following graphic illustrates the base annual prepayment rate based on the arctangent formula.



Prepayment Table

Under the Prepayment Table method, a Prepayment Table rule is referenced within the Prepayment rule for a particular product and origination date range. This prepayment table may be factored by a coefficient to scale the prepayment rates which reside in the table up or down. The prepayment table factor is also defined per product and origination date.

The Prepayment Table rule contains a table of prepayment rates dimensioned by other characteristics. The Prepayment Table rule can hold a maximum of three dimensions. For each dimension, you can define the lookup method along that dimension, either range or interpolate.

- **Range Lookup**

  Range Lookup treats the nodes within the dimension as a starting value for a range which extends to the next node dimension. For example, take an original term dimension with node values of 0, 12, and 24. The range lookup treats these values as three sets of ranges: 0 to 11, 12 to 23, and >= 24.

- **Interpolation Lookup**

  If the interpolation method is selected, the lookup applies straight line interpolation to determine the proper prepayment rate for values which fall between nodes.

- **Lookups Outside the Given Range**

  For both lookup methods, lookup for values less than the lowest node value receives the prepayment rate associated with the lowest node. Values greater than the highest node receive the prepayment rate associated with the highest node.

Along each dimension of the table, range lookup or interpolation is performed to pinpoint the proper prepayment rate from the table. Once the prepayment rate is retrieved from the prepayment table, the prepayment table factor is applied to this rate.

Base Annual Prepayment Rate = PPTableFactor * PPTableLOOKUP($dimension_x$, $dimension_y$, $dimension_z$)

3. Adjustment for Seasonality

For each prepayment method, seasonality factors can be applied to adjust the prepayment rate. The seasonality factors are defined per month. The month of the current date is used to determine the proper seasonality factor to use.

Annual Prepayment (PP) Rate = Seasonality Factor (Current Month) * Base Annual PP Rate

4. Selection of the Prepay in Full Option

If the adjusted final prepayment rate is equal to 100%, the instrument is paid off in full.

5. De-annualization of the Prepayment Option

The annual prepayment rate is adjusted to a rate per payment. The formula is as follows:

Prepay Factor = (1-(1- Annual PPRate)^(1/Payments per Year)

6. Adjustment of the Prepayment Rate for Stub or Extended Payments

The prepayment rate per payment is adjusted if the payment is a stub or extended payment. This adjustment is made in the same manner that interest cash flows are adjusted, as follows:

Adjusted prepay factor = Prepay Factor * (next payment date - last payment date) / (pmt frequency in days)

7. Determination of the Prepayment Amount

The amount of runoff due to prepayments is calculated. The prepay factor is applied to the current balance.

Prepay Runoff = Current Balance * Prepay Factor

8. Update of the Current Balance

The current balance must be reduced by the amount of prepay runoff.

Current Balance = Current Balance - Prepay Runoff

9. Application of the Prepayment Factor to Current Payment

An option exists in the Prepayment rule to reduce the payment proportionally to reflect the amount of principal that has been prepaid. If the prepayment treatment is *Refinance*, the current payment is reduced as follows:

Current Payment = Current Payment * (1 - Prepay Factor)

If the payment treatment is *Curtailment*, the current payment will remain constant, effectively reducing the term of the instrument.

Related Topics

Processing Modeling Events, page 6-13

Reprice Event

The cash flow data for the repricing event has the following characteristics:

- Static Information
- Dynamic Information
- Event Triggers

> **Important:** The modeling of an adjustable rate instrument begins at the last reprice date and ends at the next reprice date, with the next repricing date treated like a maturity date for the instrument. Therefore, no repricing events occur during a transfer pricing process.

Static Information:

- Adjustable Type Code
- Interest Rate Code
- Transfer Rate Interest Rate code
- Net Margin
- Net Margin Code
- Gross Margin
- Transfer Rate Margin
- Reprice Frequency and Multiplier
- Rate Cap Life
- Rate Floor Life
- Rate Increase Period
- Rate Decrease Period

- Rate Set Lag and Multiplier
- Rate Change Minimum
- Rate Change Rounding Code
- Rate Change Rounding Factor

Dynamic Information

- Current Gross Rate
- Current Net Rate

Event Triggers

- Bucket Start Date
- Tease End Date
- Next Reprice Date

Detail Cash Flow Data

The following table describes detail cash flow data.

Tip: The values of the Event Use codes used in the table are as follows:

- I: Initialization of record
- P: Payment
- PC: Payment recalculation
- PP: Prepayment
- R: Reprice
- D: Deferred amortization

Detail Cash Flow Data

Column Name	Column Description	Column Type	Event use
ID_NUMBER	Unique identifier	static	I
LINE_ITEM_ID	Dimension member value that determines the financial account type of detail instrument	static	I
ADJUSTABLE_TYPE _CODE	Determines whether repricing occurs, and, if it occurs, whether it occurs according to reprice dates or bucket dates	static	R
ACCRUAL_BASIS_C ODE	Method of accrual used to determine the rate per payment	static	P
AMRT_TERM & AMRT_TERM_MULT	Determines the time over which principal is amortized. Used in payment recalculation.	static	P, PC
AMRT_TYPE_CODE	Determines the method for amortizing principal. Used to match to payment pattern data.	static	PC
CUR_PAYMENT	Amount of current payment. Meaning depends on amortization type code.	dynamic	P, PC
CUR_PAR_BAL	Balance on which principal runoff, interest cash flows, and deferred runoff are based	dynamic	P, PC, PP

Column Name	Column Description	Column Type	Event use
CUR_NET_RATE	Interest rate that the financial institution pays/receives	dynamic	P, R, PC, PP
CUR_GROSS_RATE	Interest rate that the customer pays/receives. Used in determining payments and prepayments.	dynamic	P, R, PC, PP
DEFERRED_CUR_BAL	Holds current unamortized premium, discount, fees, and costs	dynamic	D
ISSUE_DATE	Date instrument is recognized as *on-the-books*		I
INT_TYPE_CODE	Determines how interest is calculated and accrued	static	P
INSTRUMENT_TYPE_CODE	Used to match a schedule instrument record to its scheduled payment dates and amounts	static	I
LAST_PAYMENT_DATE	Date of last payment before the as-of-date. Used to calculate days in first payment for interest in arrears instruments and to calculate accruals before the first payment in interest in advance instruments.	static	P
LRD_BALANCE	Balance as of last reprice date	static	I

Column Name	Column Description	Column Type	Event use
LAST_REPRICE_DATE	Date instrument rate repriced last	static	I
MATURITY_DATE	Date of final payment	static	P, PP
NEG_AMRT_AMT	Amount of current balance due to negative amortization of interest payments	dynamic	P
NEG_AMRT_EQ_DATE	Date an instrument fully re-amortizes, irrespective of payment caps	event trigger	PC
NEG_AMRT_EQ_FREQ &			
NEG_AMRT_EQ_MULT	Frequency of negative amortization equalization events. 0 denotes negative amortization equalization never occurs.	static	PC
NEG_AMRT_LIMIT	Maximum possible amount by which an instrument can negatively amortize. Stored as a percent of original balance.	event trigger	PC
NEXT_PAYMENT_DATE	Date of next payment	event trigger	P, PC
NEXT_REPRICE_DATE	Date of next rate change	event trigger	R, PP

Column Name	Column Description	Column Type	Event use
ORG_PAYMENT_AMT	Payment used for cash flow transfer pricing of fixed rate records. Used by pattern instruments to calculate payment amount.	static	PC,I
ORG_PAR_BAL	Used in conjunction with negative amortization limit to determine the maximum amount that instrument can negatively amortize. Used for Rule of 78 schedules. Used by pattern instruments to calculate payment amount.	static	PC,I
ORG_TERM & ORG_TERM_MULT	Time from origination date to maturity date. Used in determining whether an instrument balloons for payment recalculation purposes.	static	PC,PP
ORIGINATION_DATE	Determines age of instrument for prepayments. Used in calculating remaining amortization term. Used in determining payment number in pattern records.	static	PP,PC
PERCENT_SOLD	Determines net balance	static	I

Column Name	Column Description	Column Type	Event use
PMT_ADJUST_DATE	Date of next scheduled payment recalculation for negative amortization instruments	event trigger	PC
PMT_CHG_FREQ & PMT_CHG_FREQ_MULT	Frequency of regular payment change calculation for negative amortization instruments only. 0 denotes that payment never changes.	static	PC
PMT_DECR_CYCLE	Maximum possible percent decrease in payment from its previous value	static	PC
PMT_DECR_LIFE	Minimum payment amount. Stored as a percent of original payment amount. Can be overwritten on negative amortization equalization dates.	static	PC
PMT_FREQ & PMT_FREQ_MULT	Frequency of payments. Should be set equal to original term if the instrument is bullet requiring principal and interest to be paid at maturity date, or has the other asset, other liability, interest income, interest expense, non-interest income, or non-interest expense account type.	static	P, PC, PP

Column Name	Column Description	Column Type	Event use
PMT_INCR_CYCLE	Maximum possible percent increase in the value of a payment	static	PC
PMT_INCR_LIFE	Maximum payment amount. Stored as a percent of original payment amount. Can be overwritten on negative amortization equalization dates.	static	PC
RATE_CAP_LIFE	Maximum value to which current rate can reprice	static	R
RATE_CHG_MIN	Minimum amount by which current rate must change before a rate change occurs	static	R
RATE_CHG_RND_CODE	Type of rounding to be applied to current rate	static	R
RATE_CHG_RND_FAC	Precision of rounding. 0 denotes no rounding.	static	R
RATE_DECR_CYCLE	Maximum amount by which rate can decrease within a repricing period	static	R
RATE_FLOOR_LIFE	Minimum value to which current rate can fall upon repricing	static	R

Column Name	Column Description	Column Type	Event use
RATE_INCR_CYCLE	Maximum amount by which rate can increase within a repricing period	static	R
RATE_SET_LAG & RATE_SET_LAG_MULT	Time lag used when repricing. Used to determine rate set date on reprice event.	static	R
REMAIN_NO_PMTS	Number of payments left to be made on an instrument from the effective date to the maturity date	dynamic	P, PC
REPRICE_FREQ & REPRICE_FREQ_MULT	Frequency by which an instrument reprices. 0 denotes fixed rate.	static	R, PP
ORG_PAYMENT_AMT	Payment used for cash flow transfer pricing of fixed rate records	static	I
TEASER_END_DATE	Date that a teased instrument begins repricing	event trigger	R, PP
NET_MARGIN_CODE	Defines relationship between gross rate and net rate. 0 denotes floating net rate. 1 denotes constant net rate.	static	R

Related Topics

Overview of Cash Flow Calculations, page 6-1

Financial Element Calculations

The following table describes financial element calculations.

Tip: The Account Type Processing column of the table specifies for which account types financial elements are processed. The values of the codes used in the Account Type Processing column of the table are as follows:

- B: Balance only
- I: Interest only
- DCF: Detail Cash Flow
- N: Non Interest

Financial Element Calculations

Financial Element Description	Financial Element Number	Averaging Type	Weighting Factor	Account Type Processing
After Reprice Balance	255	At Last		DCF
After Reprice Gross Rate	290	At Last	After Reprice Balance	DCF
After Reprice Net Rate	300	At Last	After Reprice Balance	DCF
After Reprice Transfer Rate	310	At Last	After Reprice Balance	DCF
Average Balance	140	Daily Average		B, DCF
Average Gross Rate	150	Daily Average	Average Balance	DCF
Average Net Rate	160	Daily Average	Average Balance	DCF

Financial Element Description	Financial Element Number	Averaging Type	Weighting Factor	Account Type Processing
Average Transfer Rate	170	Daily Average	Average Balance	DCF
Before Reprice Balance	250	At First		DCF
Before Reprice Gross Rate	260	At First	Before Reprice Balance	DCF
Before Reprice Net Rate	270	At First	Before Reprice Balance	DCF
Before Reprice Transfer Rate	280	At First	Before Reprice Balance	DCF
Beginning Balance	60	At First		B, DCF
Beginning Gross Rate	70	At First	Beginning Balance	DCF
Beginning Net Rate	80	At First	Beginning Balance	DCF
Beginning Transfer Rate	90	At First	Beginning Balance	DCF
Deferred Average Balance	530	Daily Average		DCF
Deferred Ending Balance	520	At Last		DCF
Deferred Runoff	540	Accrual		DCF
Ending Balance	100	At Last		B, DCF
Ending Gross Rate	110	At Last	Ending Balance	DCF

Financial Element Description	Financial Element Number	Averaging Type	Weighting Factor	Account Type Processing
Ending Net Rate	120	At Last	Ending Balance	DCF
Ending Transfer Rate	130	At Last	Ending Balance	DCF
Fully Indexed Gross Rate	320	Sum		DCF
Fully Indexed Net Rate	330	Sum		DCF
Interest Accrual - Gross	445	Accrual		DCF, I
Interest Accrual - Net	440	Accrual		DCF, I
Interest Accrual - Transfer Rate	450	Accrual		DCF, I
Interest Cash Flow Gross	435	Sum		DCF, I
Interest Cash Flow Net	430	Sum		DCF, I
Interest Cash Flow Transfer Rate	437	Sum		DCF, I
Interest Credited	480	Sum		DCF
Lifetime Cap Balance	580	Daily Average		DCF
Lifetime Cap Effect	600	Accrual		DCF
Lifetime Cap Rate	590	Daily Average	Lifetime Cap Balance	DCF

Financial Element Description	Financial Element Number	Averaging Type	Weighting Factor	Account Type Processing
New Add Balance	340	Sum		DCF
New Add Gross Rate	350	Sum	New Add Balance	DCF
New Add Net Rate	360	Sum	New Add Balance	DCF
New Add Transfer Rate	370	Sum	New Add Balance	DCF
NGAM Balance	640	Daily Average		DCF
NGAM Interest	650	Accrual		DCF
Non-Interest Income	455	Sum		N
Non-Interest Expense	457	Sum		N
Periodic Cap Balance	550	Daily Average		DCF
Periodic Cap Effect	570	Accrual		DCF
Periodic Cap Rate	560	Daily Average	Periodic Cap Balance	DCF
Prepay Balance	515	Sum		DCF
Prepay Rate (Annual)	510	Sum	Prepay Balance	DCF
Prepay Runoff	180	Sum		DCF
Prepay Runoff	182	Sum		DCF

Financial Element Description	Financial Element Number	Averaging Type	Weighting Factor	Account Type Processing
Roll Add Balance	380	Sum		DCF
Roll Add Gross Rate	390	Sum		DCF
Roll Add Net Rate	400	Sum		DCF
Roll Add Transfer Rate	410	Sum		DCF
Scheduled Principal Runoff	190	Sum		DCF
Tease Balance	610	Daily Average		DCF
Tease Effect	630	Accrual		DCF
Tease Rate	620	Daily Average	Tease Balance	DCF
Timing of Prepay Runoff (positive)	181	Sum	Prepay Runoff (positive)	DCF
Prepay Runoff (negative)	182	Sum		DCF
Timing of Prepay Runoff (negative)	183	Sum	Prepay Runoff (negative)	DCF
Timing of Total Runoff (positive)	211	Sum	Total Runoff (positive)	DCF
Total Runoff (negative)	212	Sum		DCF
Timing of Total Runoff (negative	213	Sum	Total Runoff (negative)	DCF

Financial Element Description	Financial Element Number	Averaging Type	Weighting Factor	Account Type Processing
Total Runoff	210	Sum		DCF
Total Runoff (negative)	212	Sum		DCF
Total Runoff Gross Rate	220	Sum	Total Runoff	DCF
Total Runoff Net Rate	230	Sum		DCF
Total Runoff Transfer Rate	240	Sum		DCF
Weighted Average Term	500	Sum	Ending Balance	DCF

Related Topics

Overview of Cash Flow Calculations, page 6-1

Example of the Rule of 78

The Rule of 78 denotes an approach used by banks to formulate a loan amortization schedule. Also known as The Rule of the Sum of the Digits, this method of computing unearned interest is used on installment loans with add-on interest. The number 78 is based on the sum of the digits from 1 to 12. This approach causes a borrower to pay more interest at the beginning of the loan when there is more money owed and less interest as the obligation is reduced.

This example of the Rule of 78 is based on a 12 month loan with current payment installment of $93.33 and original balance of $1,000.00.

1. Sum all principal and interest payments made over the life of the instrument:

   ∑ Cash Flow

   = current payment * total number of payments

   = $93.33 * 12

   = $1,120.00

2. Determine total amount of interest paid over the life of the instrument.

   ∑ Interest

   = ∑ cash flow - original par balance

   = $1,120.00 - $1,000.00

   = $120.00

3. Sum the payment numbers.

   ∑ Payments

   = total num of payments * (total num of payments + 1)/2

   = 12 * 13/2

   = 78

4. Calculate principal and interest amount at each payment.

   Interest = ∑ interest * (payments remaining/∑ payments)

   Principal = current payment - interest

   The following table describes the Rule of 78 calculations for this example.

Rule of 78 Calculations

Month	Interest Calculation	Interest	Principal Calculation	Principal	Remaining Balance
1	120 * 12/78	$18.46	93.33-18.46	$74.87	$925.13
2	120 * 11/78	$16.92	93.33-16.92	$76.41	$848.72
3	120 * 10/78	$15.38	93.33-15.38	$77.95	$770.77
4	120 * 9/78	$13.85	93.33-13.85	$79.48	$691.29
5	120 * 8/78	$12.31	93.33-12.31	$81.02	$610.27
6	120 * 7/78	$10.77	93.33-10.77	$82.56	$527.71
7	120 * 6/78	$9.23	93.33-9.23	$84.10	$443.61
8	120 * 5/78	$7.69	93.33-7.69	$85.64	$357.97

Month	Interest Calculation	Interest	Principal Calculation	Principal	Remaining Balance
9	120 * 4/78	$6.15	93.33-6.15	$87.18	$270.79
10	120 * 3/78	$4.61	93.33-4.61	$88.72	$182.07
11	120 * 2/78	$3.08	93.33-3.08	$90.25	$91.82
12	120 * 1/78	$1.54	93.33-1.54	$91.79	$0.00

Related Topics

Overview of Cash Flow Calculations, page 6-1

7

Cash Flow Dictionary

This chapter lists and describes the account table columns that store cash flow data and are accessed by Oracle Transfer Pricing during cash flow processing and edits.

The chapter also provides detailed information on how to correctly populate account table columns with instrument data, including column descriptions, usage, and recommended default values for the cash flow processing columns.

This chapter covers the following topics:

- Overview of Cash Flow Columns
- Cash Flow Column Descriptions

Overview of Cash Flow Columns

The cash flow columns of the account tables store cash flow related data for financial instruments.

Oracle Transfer Pricing accesses specific columns from account tables to perform cash flow processing to generate transfer pricing and option cost results. The application also accesses account table columns to perform cash flow edits. Cash flow edit rules validate and correct, account table data used in cash flow processing. See: Overview of Cash Flow Edits Rules, *Oracle Transfer Pricing User Guide*.

The following table lists cash flow columns of account tables and provides information on whether they are associated with cash flow processing, cash flow edits, or both.

Cash Flow Columns

Column Name	Cash Flow Processing	Cash Flow Edits
ACCRUAL_BASIS_CODE	Yes	Yes

Column Name	Cash Flow Processing	Cash Flow Edits
ADJUSTABLE_TYPE_CODE	Yes	Yes
All_DIMENSION_FIELDS	Yes	No
AMRT_TERM	Yes	Yes
AMRT_TERM_MULT	Yes	Yes
AMRT_TYPE_CODE	Yes	Yes
CALENDAR_PERIOD	Yes	Yes
COMPOUND_BASIS_CODE	Yes	Yes
CUR_BOOK_BAL	Yes	Yes
CUR_GROSS_RATE	Yes	Yes
CUR_NET_PAR_BAL	No	Yes
CUR_NET_RATE	Yes	Yes
CUR_PAR_BAL	Yes	Yes
CUR_PAYMENT	Yes	Yes
CUR_TP_PER_ADB	Yes	No
DEFERRED_CUR_BAL	Yes	Yes
DEFERRED_ORG_BAL	Yes	Yes
ID_NUMBER	Yes	Yes
INSTRUMENT_TYPE_CODE	Yes	No
INT_TYPE_CODE	Yes	Yes
INTEREST_RATE_CODE	Yes	Yes

Column Name	Cash Flow Processing	Cash Flow Edits
ISSUE_DATE	Yes	Yes
LAST_PAYMENT_DATE	Yes	Yes
LAST_REPRICE_DATE	Yes	Yes
LRD_BALANCE	Yes	Yes
MARGIN	Yes	No
MARGIN_GROSS	Yes	No
MARGIN_T_RATE	Yes	No
MARKET_VALUE	Yes	No
MATCHED_SPREAD	Yes	No
MATURITY_DATE	Yes	Yes
NEG_AMRT_AMT	Yes	No
NEG_AMRT_EQ_DATE	Yes	Yes
NEG_AMRT_EQ_FREQ	Yes	Yes
NEG_AMRT_EQ_MULT	Yes	Yes
NEG_AMRT_LIMIT	Yes	Yes
NET_MARGIN_CODE	Yes	Yes
NEXT_PAYMENT_DATE	Yes	Yes
NEXT_REPRICE_DATE	Yes	Yes
ORG_BOOK_BAL	No	Yes
ORG_PAR_BAL	Yes	Yes

Column Name	Cash Flow Processing	Cash Flow Edits
ORG_PAYMENT_AMT	Yes	Yes
ORG_TERM	Yes	Yes
ORG_TERM_MULT	Yes	Yes
ORIGINATION_DATE	Yes	Yes
PERCENT_SOLD	Yes	Yes
PMT_ADJUST_DATE	Yes	Yes
PMT_CHG_FREQ	Yes	Yes
PMT_CHG_FREQ_MULT	Yes	Yes
PMT_DECR_CYCLE	Yes	Yes
PMT_DECR_LIFE	Yes	Yes
PMT_FREQ	Yes	Yes
PMT_FREQ_MULT	Yes	Yes
PMT_INCR_CYCLE	Yes	Yes
PMT_INCR_LIFE	Yes	Yes
PRIOR_TP_PER_ADB	Yes	No
PMT_SET_LAG	No	Yes
PMT_SET_LAG_MULT	No	Yes
RATE_CAP_LIFE	Yes	Yes
RATE_CHG_MIN	Yes	Yes
RATE_CHG_RND_CODE	Yes	Yes

Column Name	Cash Flow Processing	Cash Flow Edits
RATE_CHG_RND_FAC	Yes	Yes
RATE_DECR_CYCLE	Yes	Yes
RATE_DECR_LIFE	Yes	Yes
RATE_FLOOR_LIFE	Yes	Yes
RATE_INCR_CYCLE	Yes	Yes
RATE_INCR_LIFE	Yes	Yes
RATE_SET_LAG	Yes	Yes
RATE_SET_LAG_MULT	Yes	Yes
REMAIN_NO_PMTS	Yes	Yes
REMAIN_TERM_MULT	No	Yes
REPRICE_FREQ	Yes	Yes
REPRICE_FREQ_MULT	Yes	Yes
T_RATE_INT_RATE_CODE	Yes	Yes
TEASER_END_DATE	Yes	Yes
TRAN_RATE_REM_TERM	Yes	No
TRANSFER_RATE	Yes	No

Cash Flow Column Descriptions

The account table data must be clean, complete, and appropriate for Oracle Transfer Pricing to perform cash flow processing and generate accurate transfer pricing and option cost results. Although Cash Flow Edit rules clean and prepare account table data, they cannot ensure that the account table columns are populated with appropriate data that reflect business realities.

Consequently, you need the following information on correct data population of cash flow columns:

- Column name as it appears in the database (in upper case with underscores).
- Definition of the column.
- Usage in Oracle Financial Services (OFS) applications such as Oracle Transfer Pricing.
- Formulas used in the cash flow calculations.
- Data verification requirements and suggested defaults.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Accrual Basis Code (ACCRUAL_BASIS_CODE)

Definition

The basis on which the cash flow engine calculates interest accrual.

Module Usage

Oracle Transfer Pricing cash flow methodologies use ACCRUAL_BASIS_CODE for calculating interest income (financial element 430).

The following table illustrates the accrual bases and the corresponding codes, available for use by Oracle Transfer Pricing.

Accrual Bases and Corresponding Codes

Code Value	Accrual Basis
1	30/360
2	30/365
3	30/ACT
5	ACT/360

Code Value	Accrual Basis
6	ACT/365
7	ACT/ACT

Oracle Transfer Pricing cash flow methodologies reference INT_TYPE_CODE in determining whether interest payments are made in arrears or in advance.

If INT_TYPE_CODE = 1, the record is considered interest in arrears. For such a record, interest payments are paid at the end of the payment period along with the principal payments. The calculations detailed below assume that the interest is to be calculated as interest in arrears.

If INT_TYPE_CODE = 2, the record is considered interest in advance. For a description of the formula used to calculate interest in advance, see: Interest Type Code (INT_TYPE_CODE), page 7-37.

For calculation purposes, the accrual basis codes can be grouped in the following manner. Note that the calculations below assume a monthly payment frequency.

1. If the ACCRUAL_BASIS_CODE is 30/360, 30/365 or 30/Actual, the cash flow engine uses the following formula to calculate interest income on a payment date:

   *Previous Periods Ending Balance * Cur Net Rate/100 * PMT_FREQ [number of months] * x * (Next Payment Date - Last Payment Date)/(Next Payment Date - Calculated Last Payment Date)*

   Replace x with one of the three accrual basis values from above. Note that the actual denominator refers to the actual number of days in the year. Other than leap years, this equals 365 days. Also note that the Calculated Last Payment Date is the Next Payment Date rolled back by the number of months in PMT_FREQ.

   The final portion of the above calculation,

   (Next Payment Date - Last Payment Date)/(Next Payment Date Calculated Last Payment Date)

   is a ratio that calculates the percentage of the payment frequency period that should be applied to calculate the Interest Income amount. This adjustment is necessary because the Last Payment Date is not necessarily equal to the Calculated Last Payment Date for the first forecasted payment. This would be the case of a stub or extended payment at the origination or maturity of a record.

   If the Last Payment Date is precisely equal to the Calculated Last Payment Date, then the ratio will be equal to 1 and, therefore, will not impact the Interest Income calculation.

2. If the ACCRUAL_BASIS_CODE is Actual/365, Actual/Actual, or Actual/360, the

cash flow engine uses the following formula to calculate interest income on a payment date:

*Previous Periods Ending Balance * Cur Net Rate/100 * (Next Payment Date - Last Payment Date)/y*

Replace *y* with the denominator of one of the three accrual bases values from above. Note the actual numerator refers to the actual number of days in the current month.

The above two equations represent Interest in Arrears income calculations. For the interest in advance calculations, see: Interest Type Code (INT_TYPE_CODE), page 7-37.

> **Note:** If a compounding method has been chosen, the cash flow engine derives the compounded rate before calculating the interest income amounts. See Compounding Basis Code (COMPOUND_BASIS_CODE), page 7-20.

Data Verification Requirements and Suggested Defaults

- Must be equal to values 1, 2, 3, 5, 6, or 7.
- Suggested default depends on product characteristics of institution's data; default to the most common ACCRUAL_BASIS_CODE for the product.
- If AMRT_TYPE_CODE = 800, 801, or 802 (Schedule) or 1000 to 29999 (Pattern), the ACCRUAL_BASIS_CODE cannot equal 1, 2, or 3.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Adjustable Type Code (ADJUSTABLE_TYPE_CODE)

Definition

Identifies the repricing method and repricing characteristics of a record.

Module Usage

Oracle Transfer Pricing uses ADJUSTABLE_TYPE_CODE in conjunction with RERPICE_FREQ to determine the repricing characteristics of an instrument during Option Cost processing. Oracle Transfer Pricing does not use Adjustable Type Code for Standard Transfer Rate processing.

1. The following table describes the code values for this column:

Adjustable Type Code Values

Code Value	Definition	Repricing Frequency	Repricing Method
000	Fixed	0	No Repricing
030	Administered Rate	> 0	Reprices when IRC (interest rate code) changes.*
050	Floating Rate	> 0	Reprices when IRC (interest rate code) changes.*
250	Adjustable	> 0	Last Reprice Date + Reprice Frequency.*
500-998	Repricing Pattern	>0	Based on pattern definition

* If not in tease period.

1. If the ADJUSTABLE_TYPE_CODE = 0 and the REPRICE_FREQ = 0, then the record is fixed-rate.

2. If the ADJUSTABLE_TYPE_CODE = 30 or 50 and the REPRICE_FREQ > 0, then the reprice dates are driven by forecasted yield curve rate changes based on the Monte Carlo rates rather than by the REPRICE_FREQ. For these codes, the cash flow engine reprices the record by referencing the interest rate code (IRC) defined on the instrument when producing cash flow information at the beginning of each period. There is one reference to the IRC for each payment period.

   The database field, NEXT_REPRICE_DATE, is not used when the ADJUSTABLE_TYPE_CODE = 30 or 50. The database field, REPRICE_FREQ, is used to determine the yield curve point when the IRC is a yield curve as opposed to a single rate IRC.

   > **Note:** Floating/Administered ADJUSTABLE_TYPE_CODE should not be used for instruments with periodic caps or floors since periodic caps and floors infer a specific repricing frequency.

3. If the ADJUSTABLE_TYPE_CODE = 250 and the REPRICE_FREQ > 0, then the repricing frequency of the record is determined by the REPRICE_FREQ and NEXT_REPRICE_DATE. See these columns for further explanations of the repricing process.

2. When calculating Transfer Rates, the value input into the REPRICE_FREQ overrides the ADJUSTABLE_TYPE_CODE value. For instance, even though the ADJUSTABLE_TYPE_CODE > 0, if the REPRICE_FREQ = 0, the cash flow engine treats the record as a fixed-rate instrument.

> **Note:** Oracle Transfer Pricing references REPRICE_FREQ to determine if the record is adjustable.

500-998: User-Defined Repricing Patterns

Records with this range of ADJUSTABLE_TYPE_CODEs are matched to the user-defined repricing pattern in the Repricing Pattern interface. The key for matching is the ADJUSTABLE_TYPE_CODE value.

Data Verification Requirements and Suggested Defaults

- If the record is an adjustable floating or administered product, ADJUSTABLE_TYPE_CODE = 30 or 50.
- If the record reprices according to NEXT_REPRICE_DATE and REPRICE_FREQ information, ADJUSTABLE_TYPE_CODE = 250.
- If the ADJUSTABLE_TYPE_CODE > 0, REPRICE_FREQ > 0.
- If the record is fixed, ADJUSTABLE_TYPE_CODE = 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Amortization Type Code (AMRT_TYPE_CODE)

Definition

Defines the method by which the principal and interest of an account are amortized.

Module Usage

Oracle Transfer Pricing cash flow methodologies use AMRT_TYPE_CODE to determine the calculation method for the amortization of principal and interest of a record. The

following table describes the AMRT_TYPE_CODE values.

Amortization Type Code Values for Conventional Amortizing Instruments

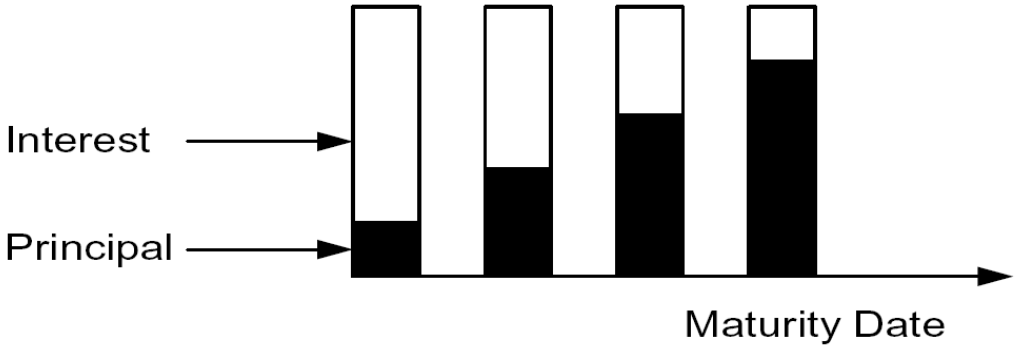
Amortization Type Code	Conventional Amortizing
100	Conventional Fixed
400	Balloon
500	Adjustable Conventional
600	Adjustable Negative Amortizing

Amortization Type Code Values for Non-conventional Amortizing Instruments

Amortization Type Code	Non-conventional Amortizing
700	Simple Interest
710	Rule of 78
800 - 802	Payment Schedules
820	Level Principal
999	Default Value
1000 -29999	User-Defined Payment Patterns

100, 400, 500, 600 - Conventionally Amortizing AMRT_TYPE_CODEs

The following graphic illustrates the amortization type of conventionally amortizing instruments.



The conventional amortization loan types have loan payments that are unevenly divided between principal balance and interest owed. Total payment amount (principal + interest) is generally equal throughout the life of the loan. The interest portion (non-shaded portion) of each payment is calculated based on the interest rate of the record and the remaining balance of the loan. Therefore, close to the loan origination, a higher portion of the payment consists of interest rather than principal. As the loan is paid, an increasing portion of each payment is allocated to principal until a zero balance is reached at maturity.

For these four AMRT_TYPE_CODEs, the amount in the CUR_PAYMENT field should equal principal and interest.

The following table describes the four conventionally amortizing amortization type codes:

Description of Conventionally Amortizing Amortization Type Codes

Conventional Loan Type	Description
100 Conventional Fixed	(Described above)
400 Balloon	A loan in which the amortization term (AMRT_TERM) of an instrument exceeds the maturity term (ORG_TERM). For example a loan with an original term of seven years is amortized conventionally as if it were a 30-year instrument. At the end of the 7th year, there exists a large balloon payment that represents 23 years of non-amortized loan balance.
500 Conventional Adjustable	Repricing instrument with conventional amortization.

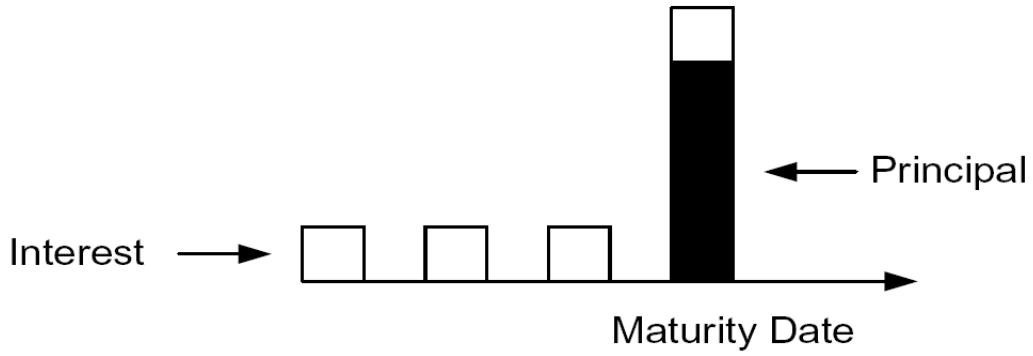
Conventional Loan Type	Description
600 Adjustable Negative Amortizing	For a negatively amortizing instrument, the principal of a loan increases when the loan payments are insufficient to pay the interest due. The unpaid interest is added to the outstanding loan balance, causing the principal to increase rather than decrease, as payments are made. See Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Note that the cash flow engine does not treat AMRT_TYPE_CODE 100, 400 or 500 differently. For a given record, the use of any of these three types produces identical results. The division is simply for product distinction purposes. For instance, AMRT_TYPE_CODE 100 can be used for a fixed-rate, adjustable-rate or a balloon record. However, only a record with an amortization type code of 600 can utilize the negative amortization fields.

- The cash flow engine does not use AMRT_TYPE_CODE to identify whether a record is adjustable or not. The cash flow engine uses REPRICE_FREQ for this purpose. Therefore, any amortization type can be adjustable.
- The cash flow engine does not use AMRT_TYPE_CODE to determine whether a record is a balloon or not. The cash flow engine uses AMRT_TERM and ORG_TERM for this purpose. Therefore, even a level principal AMRT_TYPE_CODE could be treated as a balloon instrument.
- A record must have the AMRT_TYPE_CODE 600 in order for the cash flow engine to process the record using the negative amortization fields.

700 Simple Interest (Non-amortizing)

The following graphic illustrates the Simple Interest amortization type.



For simple interest amortization type, no principal is paid until maturity. If NEXT_PAYMENT_DATE < MATURITY_DATE, the cash flow engine calculates interim interest-only payments as shown in the above diagram. The cash flow engine pays the entire principal balance for the record on the maturity date along with the appropriate interest amount.

For this AMRT_TYPE_CODE, the CUR_PAYMENT field should equal 0.

710: The Rule of 78

An amortization type in which the following calculation is used for computing the interest rebated when a borrower pays off a loan before maturity.

For example, in a 12-month loan, the total is 78 (1 + 2 + ... 12 = 78). For the first month, 12/78 of the total interest is due. In the second month, this amount is 11/78 of the total.

For Rule of 78 AMRT_TYPE_CODEs, the amount in the CUR_PAYMENT field should equal principal plus interest. See Overview of Cash Flow Calculations, page 6-1.

800 - 802: Payment Schedule

The cash flow engine matches an instrument record with its corresponding payment schedule record based on a combination of INSTRUMENT_TYPE_CODE and ID_NUMBER values.

- 800: Conventional Payment Schedule

  This AMRT_TYPE_CODE conventionally amortizes a record whose cash flows are defined in the PAYMENT_SCHEDULE table. Payment amounts contain both principal and interest.

- 801: Level Principal Payment Schedule

  This AMRT_TYPE_CODE amortizes a record whose cash flows are defined in the PAYMENT_SCHEDULE table. Payment amounts contain principal only.

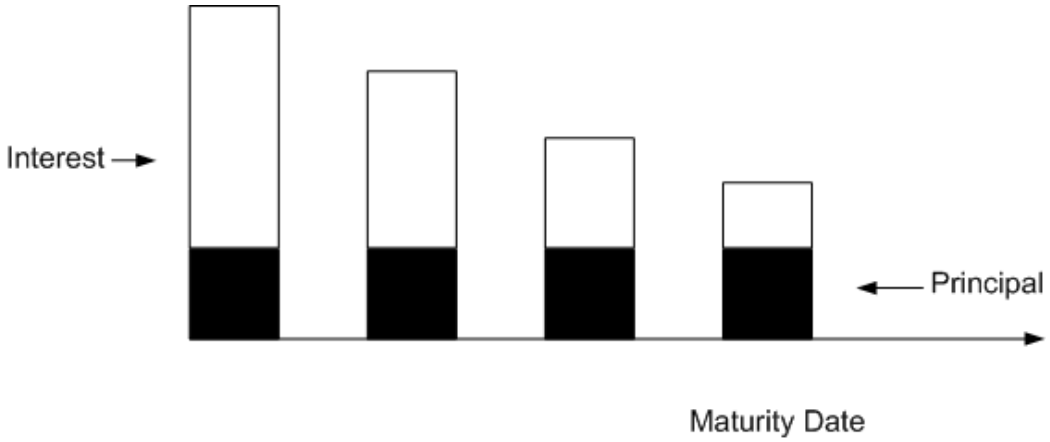
- 802: Simple Interest Payment Schedule

  This AMRT_TYPE_CODE amortizes a record whose cash flow dates are defined in the PAYMENT_SCHEDULE table. Payment amounts equal 0 (the cash flow engine

calculates interest and ignores payment amount in the schedule records). Principal balance is paid on the maturity date as defined in the schedule.

820 Level Principal Payments

The following graphic illustrates the Level Principal amortization type.



Level principal payment is the amortization type in which the principal portion of the loan payment remains constant for the life of loan. Interest (non-shaded portion) is calculated as a percentage of the remaining balance, and therefore, the interest portion decreases as the maturity date nears. Since the principal portion of payment is constant for life, the total payment amount (principal + interest) decreases as the loan approaches maturity.

For this amortization type code, the amount in the CUR_PAYMENT column should equal the principal portion only.

1000 - 29999: User-Defined Payment Patterns

Records with amortization type codes in this range are matched to the user-defined amortizations in the payment pattern interface. The key for matching is the AMRT_TYPE_CODE value. These records can only be defined as conventionally amortizing, level-principal, or simple interest.

999: Other Amortization Type

If this value is used, the payment pattern uses the AMRT_TYPE_CODE 700, simple interest amortization method. Using this AMRT_TYPE_CODE for product identification purposes is not recommended. This should be reserved for indicating erroneous data extraction.

For calculation methods of the different AMRT_TYPE_CODEs, see: Current Payment (CUR_PAYMENT), page 7-26.

Data Verification Requirements and Suggested Defaults

- All accounts require a valid AMRT_TYPE_CODE.
- If the AMRT_TYPE <> 700 or is not a simple interest-amortizing schedule or pattern record, CUR_PAYMENT must be valid. If CUR_PAYMENT value is too small or equal to 0, the cash flow engine may generate erroneous cash flows, depending on the AMRT_TYPE_CODE selected.
- If the record is defaulted to AMRT_TYPE_CODE 999, or if the cash flow engine cannot find a match in PAYMENT_SCHEDULE or PAYMENT_PATTERNS, the cash flow engine processes record as AMRT_TYPE_CODE 700.
- Suggested defaults are dependent on basic knowledge of product characteristics. The following table describes the suggested defaults:

Default Amortization Type Codes

Suggested Loan Type	AMRT_TYPE_CODE
Non-amortizing, such as Certificates of Deposit	700
Fixed amortizing, such as short term consumer loans	100
Variable amortizing, such as adjustable-rate mortgages	500

- The Rule of 78 instruments are implicitly fixed.

  If AMRT_TYPE_CODE = 710, REPRICE_FREQ = 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Amortization Term (AMRT_TERM)

Definition

Amortization term is used in conjunction with amortization term multiplier (AMRT_TERM_MULT) to define the term over which the payment is amortized.

Module Usage

AMRT_TERM is used by Oracle Transfer Pricing for adjustable rate instruments when calculating option costs.

Amortization Term

Amortization term has two main purposes:

- Identifies whether a record is a balloon and is used in calculation of payment amounts.
- Used when recalculating payment amounts for User-Defined Payment Pattern records that are defined as % Current Payment and have more than one payment frequency defined in the Payment Pattern interface.

Balloon Check

As an initial step before processing cash flows, the cash flow engine compares the ORG_TERM of a record with its AMRT_TERM. If AMRT_TERM = ORG_TERM, then the cash flow engine uses the CUR_PAYMENT from the record. The cash flow engine later recalculates the CUR_PAYMENT if:

- The record reprices.
- The TEASER_END_DATE is reached.
- A negative amortization-related recalculation date is reached.

If the AMRT_TERM > ORG_TERM, the cash flow engine recognizes the record as a balloon, and recalculates the payment amount. In order to perform this calculation, the cash flow engine must derive the remaining number of payments until the end of the amortization term. See: Current Payment (CUR_PAYMENT), page 7-26.

The remaining number of payments is calculated by adding the AMRT_TERM to the ORIGINATION_DATE to determine the amortization end date. The cash flow engine calculates the remaining number of payments by determining the number of payments that can be made between the next payment date and the amortization end date, and adding one additional payment for the payment on the next payment date. The cash flow engine uses the following formula to determine the remaining number of payments:

(((ORIGINATION_DATE - Beginning of Payment Period Date) / 30.41667 + AMRT_TERM) / PMT_FREQ)

In the formula, the Beginning of Payment Period Date refers to the date of the current payment that the cash flow engine is calculating. This would equal the NEXT_PAYMENT_DATE for the detail record if the cash flow engine were calculating the first forecasted cash flow. After the remaining number of payments are calculated, the cash flow engine derives the CUR_PAYMENT amount and applies it to the cash

flows for the record. In the absence of repricing and other recalculation events, this payment amount is paid until maturity, at which time, the cash flow engine applies the balloon payment (the remaining principal portion) for the record.

User-Defined Pattern

Records that are defined as % Current Payment in the User-Defined Payment Pattern screen and have more than one payment frequency defined in the payment pattern interface also recalculate the payment amount using the same formula.

The remaining number of payments on pattern records is calculated by rounding to the nearest number of payments when the remaining term is not exactly divisible by the payment frequency.

Data Verification Requirements and Suggested Defaults

- If the record is a balloon, AMRT_TERM > ORG_TERM.
- If the record is not a balloon, AMRT_TERM = ORG_TERM.
- AMRT_TERM should never be less than ORG_TERM.
- Do not default AMRT_TERM or ORG_TERM to 0. Use 1.
- The validation of AMRT_TERM should always be done in conjunction with AMRT_TERM_MULT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Amortization Term Multiplier (AMRT_TERM_MULT)

Definition

Used in conjunction with amortization term (AMRT_TERM) to define the term over which the payment is amortized.

Module Usage

This column is the multiplier of the AMRT_TERM column. It is used in conjunction with AMRT_TERM to define the term over which the payment is amortized. Oracle Transfer Pricing cash flow transfer-priced records reference AMRT_TERM_MULT when recalculating the current payment as defined in the AMRT_TERM column. AMRT_TERM_MULT determines the units (months, days or years) of AMRT_TERM.

Data Verification Requirements and Suggested Defaults

Values are:

D = Days

M = Months

Y = Years

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Calendar Period End Date (CAL_PERIOD_ID)

Definition

The date that the extracted data represents. Equivalent to the As of Date column in the v4.5 of Oracle Financial Services (OFS) applications.

Module Usage

CAL_PERIOD_ID is used for the following purposes:

- As a filter to identify instrument records to include in the current process.
- Transfer Pricing Remaining Term Pricing Basis.

User Preferences

Oracle Transfer Pricing uses the CAL_PERIOD_ID as a primary data filter. When executing a transfer pricing process run, the cash flow engine compares the Calendar Period defined as a Transfer Pricing Process rule run time parameter against the CAL_PERIOD_ID field of the detail account table record. If CAL_PERIOD_ID from the account table record is the same date as that from the run time parameter selection, then the cash flow engine processes the account table record. Otherwise, the cash flow engine ignores the account table record.

Transfer Pricing in Remaining Term Calculation Mode

When the Remaining Term calculation mode is selected on the Transfer Pricing rule run page, the transfer rates for the relevant methodologies are calculated from the Calendar Period date.

Data Verification Requirements and Suggested Defaults

Unless the record has a future ORIGINATION_DATE, the following conditions exist:

- CALENDAR_PERIOD = MATURITY_DATE - REMAIN TERM C.
- CALENDAR_PERIOD < NEXT_REPRICE_DATE
- CALENDAR_PERIOD < NEXT_PAYMENT_DATE
- CALENDAR_PERIOD < MATURITY_DATE
- CALENDAR_PERIOD >= ORIGINATION_DATE
- CALENDAR_PERIOD > ISSUE_DATE
- CALENDAR_PERIOD < PMT_ADJUST_DATE
- CALENDAR_PERIOD > LAST_PAYMENT_DATE

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Compounding Basis Code (COMPOUND_BASIS_CODE)

Definition

Indicates the compounding used to calculate interest income.

Module Usage

Oracle Transfer Pricing cash flow calculations reference the COMPOUND_BASIS_CODE when determining the detail records compounding method to be applied during interest income (financial element 430) calculations.

1. The following table shows the available code values for the COMPOUND_BASIS_CODE and the interest calculation logic for an annual-paying instrument with 30/360 accrual basis code.

Compounding Basis Codes: Descriptions and Annual Payment Calculations

Code Value	Description	Annual Payment Calculation
110	Daily	Balance * [(1 + Rate/365)^365-1]

Code Value	Description	Annual Payment Calculation
120	Monthly	Balance * [(1 + Rate/12)^12-1]
130	Quarterly	Balance * [(1 + Rate/4)^4-1]
140	Semiannual	Balance * [(1 + Rate/2)^2-1]
150	Annual	Balance * [(1 + Rate/1)^1-1]
160	Simple	Balance * Rate (no compounding)
170	Continuous	$e^{\text{Rate Per Payment -1}}$
200	At Maturity	Balance * Rate (no compounding)
999	Other	Balance * Rate (no compounding)

The annualized rate that is applied to the record for interest income calculations is compounded according to one of the methods listed in this table.

2. The cash flow engine compounds the rate on the record at the time of interest income calculation. If the record has repriced during option cost processing, the cash flow engine calculates the new rate, applies rounding, caps/floors, or tease periods, if any are required, and then applies the compounding calculation (COMPOUND_BASIS_CODE) before calculating interest income (financial element 430).

3. Simple and At Maturity calculate interest in the same manner. These two codes do not compound the rate.

4. Compounded interest is only calculated when the compounding frequency is less than the PMT_FREQ. If the compounding frequency is greater than the PMT_FREQ, the model assumes simple compounding.

Data Verification Requirements and Suggested Defaults

- A valid COMPOUND_BASIS_CODE; must be one of the values listed in the Compounding Basis Codes table.

- Suggested default - if the records compounding is unknown, default to COMPOUND_BASIS_CODE = 160, Simple.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Book Balance (CUR_BOOK_BAL)

Definition

Current Gross Book Balance.

Module Usage

Oracle Transfer Pricing uses CUR_BOOK_BAL in calculations for FEM_BALANCES based transfer pricing and for Charge/Credit migration processing when the Remaining Term calculation mode is selected in the Transfer Pricing Process rule.

Data Verification Requirements and Suggested Defaults

Validate that CUR_BOOK_BAL = CUR_PAR_BAL + DEFERRED_CUR_BAL.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Gross Rate (CUR_GROSS_RATE)

Definition

Coupon rate of account, expressed in terms of an annualized rate.

Module Usage

When the Model with Gross Rates option is selected while setting the Transfer Pricing rule and Remaining Term calculation mode is selected on the Transfer Pricing Process rule, CUR_GROSS_RATE is used by the cash flow engine. When this option is selected, the cash flow engine uses the CUR_GROSS_RATE of the records for the following two calculations:

Amortization

When conventionally amortizing a records balance with the Model with Gross Rates

option selected, the cash flow engine uses the CUR_GROSS_RATE as the customer rate. If the option is not selected, then the cash flow engine uses CUR_NET_RATE as the customer rate for amortization calculation.

Prepayments

To determine the prepayment rate, the current customer rate must be compared to the market rate. If the Model with Gross Rates option is selected, then the customer rate is represented by the CUR_GROSS_RATE. If the Model with Gross Rates option is not selected, the CUR_NET_RATE is used as the current customer rate.

If Model with Gross Rates is selected, the cash flow engine uses the CUR_GROSS_RATE for gross interest cash flow (financial element 435) calculations. This means that the record amortizes and prepays according to the CUR_GROSS_RATE, but the net cash flows associated interest income (financial element 430) is calculated from the CUR_NET_RATE.

> **Note:** Depending on the NET_MARGIN_CODE value, interest income is calculated differently.

For a complete explanation of the relationship between NET_MARGIN_CODE, CUR_GROSS_RATE, and CUR_NET_RATE, see: Net Margin Code (NET_MARGIN_CODE), page 7-52.

Data Verification Requirements and Suggested Defaults

- All term accounts require a valid CUR_GROSS_RATE.
- For non-interest earning/bearing accounts, CUR_GROSS_RATE = 0.
- For transaction accounts where the rate changes daily, based upon average balances, CUR_GROSS_RATE should be the spot rate at the time the extract program is run.
- CUR_GROSS_RATE >= 0.
- CUR_GROSS_RATE = MARGIN_GROSS + value of index (IRC) that the account is tied to (assuming periodic/lifetime caps/floors, tease periods do not apply and rounding is taken into consideration).
- CUR_GROSS_RATE should be validated while validating CUR_PAYMENT. For validation formulas, see: Current Payment (CUR_PAYMENT), page 7-26.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Net Rate (CUR_NET_RATE)

Definition

Interest rate on which interest income due to the bank is based.

Module Usage

When the Remaining Term calculation mode is selected on the Transfer Pricing Process rule run page, Oracle Transfer Pricing cash flow calculations reference CUR_NET_RATE for the following purposes:

Interest Income (Financial Element 430) Calculation

CUR_NET_RATE is used to derive the interest cash flow (income/expense) that is due to the financial institution (referred to as net). The cash flow engine uses different interest income calculations depending on the ACCRUAL_BASIS_CODE and INT_TYPE. These calculations are presented under the Accrual Basis Code (ACCRUAL_BASIS_CODE) and Interest Type Code (INT_TYPE_CODE). Interest income is calculated on payment dates or the records maturity date. As the calculations indicate, after referencing the ACCRUAL_BASIS_CODE, the cash flow engine applies the CUR_NET_RATE to the entire payment period (last Previous payment date to next Current payment date). If any repricing occurred during the payment period, the cash flow engine uses the last repriced rate that occurred immediately before the next Current payment date.

Prepayments

As defined in the Prepayment rule interface, the cash flow engine compares the customer rate to the market rate when determining the prepayment rate. If the Model with Gross Rates option is not selected, the CUR_NET_RATE is the customer rate and therefore is used for prepayment calculations.

Amortization

When amortizing the balance of a record, a key input is customer rate for the record. If the Model with Gross Rates options is not selected, then the cash flow engine uses the CUR_NET_RATE for amortization purposes.

> **Note:** Depending on the NET_MARGIN_CODE value, interest income is calculated differently. For a complete explanation of the relationship between NET_MARGIN_CODE, CUR_GROSS_RATE, and CUR_NET_RATE, see: Net Margin Code (NET_MARGIN_CODE), page 7-52.

Data Verification Requirements and Suggested Defaults

- All term accounts require a valid CUR_NET_RATE.
- For non-interest earning/bearing accounts, CUR_NET_RATE = 0.
- For transaction accounts where the rate changes daily based upon average balances, CUR_NET_RATE should be the spot rate at the time the extract program is run.
- For interest-bearing accounts, CUR_NET_RATE >= 0.
- CUR_NET_RATE = MARGIN + value of index that the account is tied to (assuming periodic/lifetime caps/floors do not apply and rounding is taken into consideration).

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Option-Adjusted Spread (CUR_OAS)

Definition

The average spread over all stochastic rate paths that equate the discounted sum of future cash flows to the target balance at the Calendar Period date.

Module Usage

When the Remaining Term calculation mode is selected in the Transfer Pricing Process Rule, the Oracle Transfer Pricing option cost process writes the result of its option-adjusted spread calculations to this column.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Par Balance (CUR_PAR_BAL)

Definition

Represents the starting balance from which remaining term cash flows are generated.

Module Usage

Oracle Transfer Pricing Cash Flow Remaining Term Pricing Basis methodologies use

the CUR_PAR_BAL field to derive the starting balance for amortization calculations. For amortizing accounts, the cash flow engine amortizes the Current Par Balance (CUR_PAR_BAL) over the remaining number of payments.

1. For the cash flow Remaining Term Pricing Basis methodologies in Oracle Transfer Pricing, as the cash flow engine processes the payment dates and the maturity date for the record, the CUR_PAR_BAL is reduced by the principal portion of the CUR_PAYMENT amount until the principal balance reaches 0. Once the balance has been reduced to 0, processing of the record ceases. The calculation method that defines how the CUR_PAR_BAL amount is reduced is represented by the AMRT_TYPE_CODE and the CUR_PAYMENT fields.

2. For User-Defined Payment Patterns where the payment method is defined as "% Current Balance", Oracle Transfer Pricing references the CUR_PAR_BAL column for all payment amounts including the first one.

Data Verification Requirements and Suggested Defaults

- CUR_PAR_BAL requires a valid balance for all accounts. If CUR_PAR_BAL = 0, the Cash Flow Engine will not process the record.
- CUR_PAR_BAL = CUR_BOOK_BAL - DEFERRED_CUR_BAL
- CUR_PAR_BAL should have the same sign as CUR_PAYMENT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Payment (CUR_PAYMENT)

Definition

Represents the current periodic payment made to repay the outstanding balance.

Module Usage

For standard amortization types (those that are not nonpatterned and nonscheduled), Oracle Transfer Pricing (remaining term methods) uses the CUR_PAYMENT from the detail record for the life of the record until a payment recalculation occurs. A payment recalculation occurs when the record is an:

- Adjustable record and a reprice date (NEXT_REPRICE_DATE) is reached.
- Adjustable record and the TEASER_END_DATE is reached (note that if TEASER_END_DATE < NEXT_REPRICE_DATE, TEASER_END_DATE takes

precedence).

- AMRT_TYPE = 600 and the PMT_ADJUST_DATE, NEG_AMRT_EQ_DATE, or the NEG_AMRT_LIMIT is reached.

Depending on AMRT_TYPE_CODE, CUR_PAYMENT may be composed of principal or interest or both.

AMRT_TYPE_CODE 700 (Simple Interest):

CUR_PAYMENT equals interest only. The cash flow engine always calculates the interest component of any payment amount. Therefore, for extracting purposes, for a simple interest record CUR_PAYMENT = 0.

AMRT_TYPE_CODE 100, 400, 500, 600 (Conventionally Amortizing):

For extracting purposes, CUR_PAYMENT = principal + interest.

For additional information on AMRT_TYPE_CODE 600, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

AMRT_TYPE_CODE 820 (Level Principal):

For extracting purposes, CUR_PAYMENT = principal only.

- **AMRT_TYPE_CODE 800:** Conventional payment schedule. Payment Amount contains both principal and interest.

- **AMRT_TYPE_CODE 801:** Level principal payment schedule. Payment Amount contains principal only.

- **AMRT_TYPE_CODE 802:** Simple interest payment schedule. Payment Amount equals zero (for simple interest, the cash flow engine ignores this field and just looks at the scheduled payment date).

AMRT_TYPE_CODE 1000 - 29999, User-Defined Payment Patterns:

Note: Oracle Transfer Pricing does not reference CUR_PAYMENT when using the User-Defined Payment Patterns.

Data Verification Requirements and Suggested Defaults

- If AMRT_TYPE_CODE = 100, 400, 500, 600, 710, 800 or is a conventionally-amortizing payment pattern, CUR_PAYMENT includes principal and interest.

- If AMRT_TYPE_CODE = 820, 801 or is a level principal-amortizing payment pattern, CUR_PAYMENT includes principal only.

- If AMRT_TYPE_CODE = 700, 802, or is a simple interest-amortizing payment pattern, CUR_PAYMENT can be 0.

- For AMRT_TYPE_CODE <> Simple Interest AMRT_TYPE_CODEs, CUR_PAYMENT must have the same sign as CUR_BOOK_BAL.

- CUR_PAYMENT must have the same sign as the CUR_BOOK_BAL and CUR_PAR_BAL columns.

- If AMRT_TYPE_CODE = 600 (Negative Amortization) and PMT_DECR_LIFE <> 0, CUR_PAYMENT is greater than or equal to ORG_PAYMENT_AMT * (1 - PMT_DECR_LIFE/100).

- If AMRT_TYPE_CODE = 600 (Negative Amortization) and PMT_INCR_LIFE <> 0, CUR_PAYMENT is lesser than or equal to ORG_PAYMENT_AMT * (1 + PMT_INCR_LF/100).

- CUR_PAYMENT can be validated by performing the following calculations:

  - For conventionally amortizing and Rule of 78:

    CUR_PAYMENT = (CUR_BOOK_BAL* (CUR_GROSS_RATE/((12/PMT_FREQ[in months]) *100))) / (1 - ((1+(CUR_GROSS_RATE /((12/ PMT_FREQ [in months])*100))) ^-(REMAIN_NO_PMTS)))

  - For fixed rate accounts:

    CUR_PAYMENT = (ORG_BOOK_BAL* (CUR_GROSS_RATE/((12/PMT_FREQ[in months]) *100))) / (1 - ((1+(CUR_GROSS_RATE /((12/ PMT_FREQ [in months])*100))) ^-(ORG_TERM/PMT_FREQ [in months])))

  - For Level Principal records:

    CUR_PAYMENT = CUR_BOOK_BAL / REMAIN_NO_PMTS

  - For fixed-rate accounts level principal records, the following should also be true:

    CUR_PAYMENT = ORG_BOOK_BAL/ (ORG_TERM/ PMT_FREQ [in months])

  - For balloon records, the calculated remaining number of payments in the amortization term (CRPAT) must be calculated first. See: Remaining Number of Payments (REMAIN_NO_PMTS), page 7-86.

    The following calculation is used:

    CUR_PAYMENT = (CUR_BOOK_BAL* (CUR_GROSS RATE/((12/PMT_FREQ[in months]) *100))) / (1 - ((1+(CUR_GROSS_RATE /((12/

PMT_FREQ [in months])*100))) ^-(CRPAT)))

- For fixed rate accounts, the following should be true:

  CUR_PAYMENT = (ORG_BOOK_BAL* (CUR_GROSS_RATE/((12/PMT_FREQ[in months]) *100))) / (1 - ((1+(CUR_GROSS_RATE /((12/ PMT_FREQ [in months])*100))) ^-(AMRT_TERM/PMT_FREQ [in months])))

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Static Spread (CUR_STATIC_SPREAD)

Definition

The spread over the implied forward rates that equates the discounted sum of future cash flows to the target balance at the Calendar Period.

Module Usage

When Remaining Term calculation mode is selected in the Transfer Pricing Process Rule, the Oracle Transfer Pricing option cost process writes the result of its static spread calculations to this column.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Current Transfer Pricing Period Average Daily Balance (CUR_TP_PER_ADB)

Definition

The average balance at the LAST_REPRICE_DATE.

Module Usage

When processing with the mid-period repricing option, Oracle Transfer Pricing references CUR_TP_PER_ADB as the average daily balance at the time of the last repricing event. This field is used in conjunction with the PRIOR_TP_PER_ADB field.

1. Mid-period repricing produces an average transfer rate over the current processing month if the LAST_REPRICE_DATE occurred since the beginning of the processing

month. CUR_TP_PER_ADB and PRIOR_TP_PER_ADB are used as average balance weightings in the mid-period pricing equation.

The following table presents a mid-period repricing scenario, which is followed by an explanation.

A Sample Mid-Period Repricing Scenario

Field	Scenario
CALENDAR_PERIOD	12/31/2006
LAST_REPRICE_DATE	12/15/2006
LAST_PAYMENT_DATE	12/15/2006 (balance was reduced on this date)
CUR_TP_PER_ADB	$10,000
PRIOR_TP_PER_ADB	$15,000
TRANSFER_RATE	3% from 11/15/2006 to 12/15/2006 (prior period, 30 days in the period)
TRANSFER_RATE	5% from 12/15/2006 to 1/15/2007 (current period, 31 days in the period)

Without the mid-period repricing option, the cash flow engine would assign a 5% transfer rate to the record for the month of December. However, this is the transfer rate only for the second half of December. The true transfer rate for the month should be a balance-weighted average transfer rate over the entire month. Mid-period repricing provides this by calculating the final transfer rate by weighting the transfer rate results (from current and previous repricing periods) by average balances and days. This final transfer rate is then applied to the detail records TRANSFER_RATE field.

The equation used by Oracle Transfer Pricing for calculating Mid-Period Repricing is as follows:

((CUR_TP_PER_ADB * Current Period Transfer Rate * Current Period Days) + (PRIOR_TP_PER_ADB * Prior Period Transfer Rate * Prior Period Days)) / ((CUR_TP_PER_ADB * Current Period Days) + (PRIOR_TP_PER_ADB * Prior Period Days))

From the example from above, the equation would be:

((10,000 * 5% * 31) + (15,000 * 3% * 30))/ ((10,000 * 31) + (15,000 * 30)) = 3.82%

Therefore, the correct transfer rate is 3.82%.

2. In reference to this calculation, the CUR_TP_PER_ADB is used to determine the balance as of the LAST_REPRICE_DATE and PRIOR_TP_PER_ADB is used to determine the balance as of the repricing dates prior to the LAST_REPRICE_DATE.

3. If the TEASER_END_DATE is greater than the CALENDAR_PERIOD, the mid-period repricing does not apply and the CUR_TP_PER_ADB and PRIOR_TP_PER_ADB columns are not used.

Data Verification Requirements and Suggested Defaults

- If the record is adjustable and repricing occurs within the month, CUR_TP_PER_ADB = (average) balance at the time of the LAST_REPRICE_DATE.

- If the CUR_TP_PER_ADB and PRIOR_TP_PER_ADB are not available, use CUR_PAR_BAL as your default (otherwise mid-period repricing could result in a zero transfer rate).

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Deferred Current Balance (DEFERRED_CUR_BAL)

Definition

Current non-amortized deferred balance representing future income/expense such as premium, discount, fees, and costs.

Module Usage

DEFERRED_CUR_BAL holds the discount or premium (fee or cost) associated with a bond or loan record.

Discounted Instrument

A discount loan or instrument is one with interest deducted from the face value (CUR_PAR_BAL) of the loan at its origination. The discount amount (DEFERRED_CUR_BAL) is the difference between the loans current market price (CUR_BOOK_BAL) and its stated par value (CUR_PAR_BAL). The DEFERRED_CUR_BAL, which represents income, will be amortized over the life of the instrument according to a constant yield calculation. Therefore, as the instrument approaches maturity, the CUR_BOOK_BAL approaches the CUR_PAR_BAL.

For discounted instruments, the DEFERRED_CUR_BAL is a negative balance. This

indicates to the cash flow engine that the balance is income.

Premium Instrument

A premium bond or instrument is one in which the face value is issued below the book value. The premium is represented by the DEFERRED_CUR_BAL field, and, as with discounts, the deferred portion is accreted over the life of the instrument. For a premium instrument the DEFERRED_CUR_BAL represents an expense.

For instruments sold at a premium, DEFERRED_CUR_BAL is positive, indicating that the balance is an expense.

The relationship in the cash flow engine between book, par, and the deferred amount is as follows:

CUR_BOOK_BAL = CUR_PAR_BAL + DEFERRED_CUR_BAL

Given below is an example of this relationship for a discounted loan:

CUR_PAR_BAL = $9,000

CUR_BOOK_BAL = $8,000

DEFERRED_CUR_BAL = $1,000

1. The cash flow engine performs a constant-yield amortization of the DEFERRED_CUR_BAL. This allows the deferred balance to be accreted evenly over the life of the instrument. This lifelong accretion rather than a one-time realization of the deferred amount at the inception of the instrument is dictated by general accounting rules regarding discount or premium instruments. For certain fees and costs, as well as premiums and discounts, banks must recognize income/expense over the life of an account instead of at the inception of the account. Hence, some deferred balances are amortized over the maturity term of an account even if the account itself does not amortize. See: Detail Cash Flow Data, page 6-35.

2. If the DEFERRED_CUR_BAL = 0, the cash flow engine will recognize this record as having no discount or premium (CUR_BOOK_BAL = CUR_PAR_BAL).

Data Verification Requirements and Suggested Defaults

- For deferred income (fees or discount) DEFERRED_CUR_BAL < 0.
- For deferred expense (costs or premium) DEFERRED_CUR_BAL > 0.
- For accounts with deferred balances, the following equation must hold true: CUR_BOOK_BAL = CUR_PAR_BAL + DEFERRED_CUR_BAL
- For accounts with no deferred balances, DEFERRED_CUR_BAL = 0

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Deferred Original Balance (DEFERRED_ORG_BAL)

Definition

Original non-amortized deferred balance representing future income/expense such as premium, discount, fees, and costs.

Module Usage

This column must exist in the account table for cash flow processing, but is not used in any of the Transfer Pricing calculations.

Data Verification Requirements and Suggested Defaults

Not applicable.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Gross Margin (MARGIN_GROSS)

Definition

The contractual spread that is added to the pricing index, which results in the customer (Gross) rate, for adjustable rate accounts.

Module Usage

MARGIN_GROSS is used by Oracle Transfer Pricing during option cost processing when Model with Gross rates option has been selected while defining the Transfer Pricing rule.

Data Verification Requirements and Suggested Defaults

- For fixed rate accounts, MARGIN_GROSS = 0.
- For adjustable rate accounts with no contractual margin, MARGIN_GROSS = 0.
- For administered rate accounts, MARGIN_GROSS = 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Historic Option-Adjusted Spread (HISTORIC_OAS)

Definition

The average spread over all stochastic rate paths that equates the discounted sum of future cash flows to the target balance at origination.

Module Usage

When Standard calculation mode is selected in the Transfer Pricing Process Rule, the Oracle Transfer Pricing option cost process writes the result of its option-adjusted spread calculations to this column.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Historic Static Spread (HISTORIC_STATIC_SPREAD)

Definition

The spread over the implied forward rates that equates the discounted sum of future cash flows to the target balance at origination.

Module Usage

When Standard calculation mode is selected in the Transfer Pricing Process rule, the Oracle Transfer Pricing option cost process writes the result of its static spread calculations to this column.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

ID Number (ID_NUMBER)

Definition

Account number identifying individual customer accounts.

Module Usage

ID_NUMBER identifies the individual customer accounts in account tables. The ID_NUMBER should be unique for a given Calendar Period within an account table. The cash flow engine utilizes ID_NUMBER to identify an account as it is processed.

It is also important for instruments with payment schedules (AMRT_TYPE_CODE 800, 801, 802) because the cash flow engine uses the combination of INSTRUMENT_TYPE_CODE and ID_NUMBER to determine the payment dates and amounts from the PAYMENT_SCHEDULE table.

Data Verification Requirements and Suggested Defaults

- ID_NUMBER is loaded into account tables from the source data. Because the detail client data loader ensures that ID_NUMBER is unique for each Calendar Period, there are no edits or defaults for this field.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Instrument Type Code (INSTRUMENT_TYPE_CODE)

Definition

Code identifying the instrument category of the customer account. This code generally matches the Account Table in which the data is loaded.

Module Usage

INSTRUMENT_TYPE_CODE identifies the instrument category of an account. The following table lists each of the available INSTRUMENT_TYPE_CODE values:

Instrument Type Codes and Descriptions

Code Value	Description
110	Commercial loans
120	Consumer loan
130	Mortgages
140	Investments
141	MBS
150	Credit card
210	Deposits
220	Wholesale funding

The cash flow engine utilizes the INSTRUMENT_TYPE_CODE to determine the instrument of an account when accounts of different instrument category are grouped together (on a report or other query).

It is also important for instruments with Payment Schedules (AMRT_TYPE_CODE 800, 801, 802) because the cash flow engine uses the INSTRUMENT_TYPE_CODE and ID_NUMBER to determine the payment dates and amounts from PAYMENT_SCHEDULE table.

Data Verification Requirements and Suggested Defaults

The INSTRUMENT_TYPE_CODE value for an individual account should match the instrument type of the table in which it is stored. For example, all account records stored in the DEPOSITS table should be assigned INSTRUMENT_TYPE_CODE = 210.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Interest Type Code (INT_TYPE_CODE)

Definition

Determines whether interest cash flows are paid in advance or in arrears.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference INT_TYPE_CODE to determine whether interest payments are made in arrears or in advance. INT_TYPE_CODE impacts the calculation of interest income (financial element 430, 435).

1. If INT_TYPE_CODE = 1, the record is considered interest in arrears. Interest payments are paid at the end of the payment period along with the principal payments. See: Accrual Basis Code (ACCRUAL_BASIS_CODE), page 7-6.

2. If INT_TYPE_CODE = 2, the record is considered interest in advance. Interest payments are paid at the beginning of the payment period starting from the ORIGINATION_DATE. Payments are made on every payment date except for the MATURITY_DATE.

3. The calculation used to determine interest income (financial element 430) for an interest in advance record depends also on the ACCRUAL_BASIS_CODE. The equations relevant to an interest in advance calculation are as follows:

   For ACCRUAL_BASIS_CODE 30/360, 30/365 and, 30/Actual the interest income calculation, when PMT_FREQ_MULT = M (assuming no compounding), is:

   Current Periods Ending Balance * Cur Net Rate/100 * PMT_FREQ [number of months] * [accrual basis] * (Following Payment Date - Next Payment Date)/(Calculated Following Payment Date - Next Payment Date)

   Here:

   - Following Payment Date is the payment after Next Payment Date - (a).

   - Calculated Following Payment Date is the Next Payment Date rolled forward by the number of months in PMT_FREQ -(b).

     In most cases, (a) is the same as (b). However, if there is a short or extended maturity, (a) <> (b), and therefore the cash flow engine needs to consider this factor while calculating the last interest cash flow (in other words, the payment just before maturity).

     > **Note:** The Following Payment Date is the payment that follows the one currently being calculated.

For ACCRUAL_BASIS_CODE Actual/365, Actual/Actual, Actual/360 (the example is for an Actual/365 record), the interest income calculation is:

Current Periods Ending Balance * Cur Net Rate/100 * (Following Payment Date - Current Payment Date)/365

4. Even though the cash flow engine pays interest in advance on every payment date except for the MATURITY_DATE, the REMAIN_NO_PMTS column should count MATURITY_DATE as a payment date since principal is still paid on this date.

Data Verification Requirements and Suggested Defaults

- If INT_TYPE_CODE = 2, AMRT_TYPE_CODE = 700, 820, 801, 802 or non-conventionally amortizing User-Defined Payment Patterns.
- INT_TYPE_CODE valid values are 1 and 2.
- If INT_TYPE_CODE = 2, REMAIN_NO_PMTS still counts MATURITY_DATE as a payment date.
- If INT_TYPE_CODE = 2 and ORIGINATION_DATE > CALENDAR_PERIOD, NEXT_PAYMENT_DATE and LAST_PAYMENT_DATE = ORIGINATION_DATE.
- If the ORIGINATION_DATE > CALENDAR_PERIOD, the NEXT_PAYMENT_DATE and LAST_PAYMENT_DATE both equal the ORIGINATION_DATE.
- For conventionally amortizing records, interest in advance is not a valid INT_TYPE_CODE. Interest in advance functions with simple interest and level principal AMRT_TYPE_CODEs.
- If a compounding method has been chosen, the cash flow engine derives the compounded rate before calculating the interest income amounts. See: Compounding Basis Code (COMPOUND_BASIS_CODE), page 7-20.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Interest Rate Code (INTEREST_RATE_CODE)

Definition

Identifies the index to which adjustable rate accounts are tied.

Module Usage

Oracle Transfer Pricing references INTEREST_RATE_CODE when calculating option costs and uses the relevant interest rate from the defined interest rate code (IRC) together with the margin to determine the reprice rate for the instrument. The value populated in this column for each account record must be synchronized with the actual interest rate code values defined in Oracle Transfer Pricing.

Data Verification Requirements and Suggested Defaults

- If ADJUSTABLE_TYPE_CODE = 0, INTEREST_RATE_CODE can be defaulted to 0.
- If ADJUSTABLE_TYPE_CODE <> 0, INTEREST_RATE_CODE = 1 - 99999.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Issue Date (ISSUE_DATE)

Definition

Date the account was originated (issued) by the originating institution.

Module Usage

Oracle Transfer Pricing does not reference Issue Date. However, it is recommended that you populate this field with the actual issue date or as a default equal to ORIGINATION_DATE.

Data Verification Requirements and Suggested Defaults

- For accounts originated by the current institution, ISSUE_DATE = ORIGINATION_DATE.
- For accounts acquired through acquisition of another institution, or purchase of a pool of accounts, ISSUE_DATE < ORIGINATION_DATE.
- If ORIGINATION_DATE from the original institution is not available, ISSUE_DATE = ORIGINATION_DATE.
- ISSUE_DATE + ISSUE TERM = MATURITY_DATE
- ISSUE_DATE <= LAST_REPRICE_DATE

- ISSUE_DATE <= ORIGINATION_DATE. ISSUE_DATE cannot be greater than ORIGINATION_DATE.

- For adjustable-rate records, when the LAST_REPRICE_DATE is less than the ISSUE_DATE, transfer pricing does not occur. If ADJUSTABLE_TYPE_CODE = 0, LAST_REPRICE_DATE can be defaulted to be less than or equal to the ISSUE_DATE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Last Payment Date (LAST_PAYMENT_DATE)

Definition

The date on which the last payment for the record was made.

Module Usage

When Remaining Term calculation mode is selected on the Transfer Pricing Process rule run page Oracle Transfer Pricing uses LAST_PAYMENT_DATE for cash flow transfer pricing methods to calculate the payment period and interest income (financial element 430) for the first cash flow.

1. The first cash flow from the CALENDAR_PERIOD references NEXT_ PAYMENT_DATE minus LAST_PAYMENT_DATE in order to determine the payment period for interest income calculations. The use of LAST_PAYMENT_ DATE rather than (NEXT_PAYMENT_DATE - PMT_FREQ) allows for short or extended first period payments. Beyond the first forecasted cash flow (NEXT_ PAYMENT_DATE), the cash flow engine rolls forward by PMT_FREQ until MATURITY_ DATE. See: Next Payment Date (NEXT_PAYMENT_DATE), page 7-53.

2. For instruments that have been originated in the past (CALENDAR_PERIOD >= ORIGINATION_DATE), the LAST_PAYMENT_DATE is always greater than or equal to the ORIGINATION_DATE.

3. For future originations (CALENDAR_PERIOD < ORIGINATION_DATE), LAST_ PAYMENT_DATE is always equal to the ORIGINATION_DATE.

4. Even though the first cash flow may be extended, the PMT_FREQ is always extracted as the general frequency of payment for the record.

5. For interest income calculation examples that reference LAST_PAYMENT_ DATE,

see:

- Accrual Basis Code (ACCRUAL_BASIS_CODE), page 7-6.
- Interest Type Code (INT_TYPE_CODE), page 7-37.

Data Verification Requirements and Suggested Defaults

- Default to NEXT_PAYMENT_DATE - PMT_FREQ if actual LAST_PAYMENT_DATE is not available.
- LAST_PAYMENT_DATE < NEXT_PAYMENT_DATE.
- If ORIGINATION_DATE > CALENDAR_PERIOD, LAST_PAYMENT_DATE = ORIGINATION_DATE.
- If ORIGINATION_DATE <= CALENDAR_PERIOD, LAST_PAYMENT_DATE >= ORIGINATION_DATE.
- If INT_TYPE = 2 and ORIGINATION_DATE > CALENDAR_PERIOD, NEXT_PAYMENT_DATE and LAST_PAYMENT_DATE = ORIGINATION_DATE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Last Repricing Date (LAST_REPRICE_DATE)

Definition

For adjustable rate accounts, the last date that the current interest rate changed.

Module Usage

1. For an adjustable-rate straight term transfer-priced record, Oracle Transfer Pricing uses the LAST_REPRICE_DATE to identify the assignment date (Yield Curve Date). The assignment date of the Interest Rate Code just before (or equal to) the LAST_REPRICE_DATE of the record is used as the transfer pricing yield curve.

   For example, if the LAST_REPRICE_DATE of the record is 1/15/2006 and the Historical Rate interest rate code (IRC) is defined at monthly intervals and only at month-end, the assignment date would be 12/31/2005. The REPRICE_FREQ is then matched to the same term on the transfer pricing yield curve (IRC) defined in the Historical Rates in Oracle Transfer Pricing.

2. For an adjustable-rate cash flow transfer-priced record, Oracle Transfer Pricing uses cash flow methods to transfer price all payments that occur from the LAST_REPRICE_DATE to the NEXT_REPRICE_DATE. In this case, the term and date as defined by these two columns are not used directly to define the transfer rate. They are the starting and ending points within which the cash flow engine applies cash flow transfer pricing.

3. For fixed-rate records, the LAST_REPRICE_DATE and NEXT_REPRICE_DATE are not referenced. ORIGINATION_DATE and MATURITY_DATE are used instead.

Data Verification Requirements and Suggested Defaults

- If REPRICE_FREQ <> 0, LAST_REPRICE_DATE <= CALENDAR_PERIOD.
- If REPRICE_FREQ <> 0 and TEASER_END_DATE <= CALENDAR_PERIOD, LAST_REPRICE_DATE + REPRICE_FREQ = NEXT_REPRICE_DATE.
- LAST_REPRICE_DATE < NEXT_REPRICE_DATE.
- If REPRICE_FREQ = 0, LAST_REPRICE_DATE = ORIGINATION_DATE.
- If REPRICE_FREQ <> 0, LAST_REPRICE_DATE >= ORIGINATION_DATE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Last Reprice Date Balance (LRD_BALANCE)

Definition

Balance as of the previous repricing event of the record.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference LRD_BALANCE when transfer-pricing adjustable-rate records. LRD_BALANCE holds the balance as of the LAST_REPRICE_DATE.

When transfer pricing adjustable-rate records, LRD_BALANCE is used as the starting balance from the LAST_REPRICE_DATE. When cash flow transfer pricing an adjustable-rate record, the cash flow engine calculates the payment events from the LAST_REPRICE_DATE to the NEXT_REPRICE_DATE. If payments (amortization) occurred in between the LAST_REPRICE_DATE and the CALENDAR_PERIOD, CUR_PAR_BAL of the record is smaller than it was on the LAST_REPRICE_DATE. Therefore, in order to provide an accurate balance amount at the time of the

LAST_REPRICE_DATE, the LRD_BALANCE has been provided. Oracle Transfer Pricing amortizes the LRD_BALANCE from the LAST_REPRICE_DATE until the NEXT_REPRICE_DATE.

Data Verification Requirements and Suggested Defaults

- If the record is fixed-rate, LRD_BALANCE = ORG_PAR_BAL.
- If the record is adjustable-rate, LRD_BALANCE = balance as of the last reprice date.
- If the balance as of the last reprice date is not known, default LRD_BALANCE to CUR_PAR_BAL.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Margin (MARGIN)

Definition

MARGIN is the contractual spread that is added to the pricing index and results in the financial institutions retention (net) rate, for adjust-able-rate accounts.

Module Usage

1. MARGIN is used during cash flow generation when calculating option costs for adjustable rate instruments.
2. For adjustable-type records, MARGIN is the contractual spread above/below the index that is applied throughout the life of the instrument. The financial institutions retention rate (CUR_NET_RATE) is equal to the index that the record is tied to plus a spread, which is defined by the MARGIN field.
3. The events of a repricing involving MARGIN are as follows: At a repricing event (or a TEASER_END_DATE) for an adjustable-rate record, the cash flow engine matches the INTEREST_RATE_CODE, REPRICE_FREQ and repricing date of the detail record to the rate generated by the Monte Carlo rate engine. After matching the rate, the cash flow engine adds the MARGIN amount and apply any teases, rate caps/floors, and rounding to derive the rate that will be applied to the record.

   Note: Note: The cash flow engine does not reference repricing date information for ADJUSTABLE_TYPE_CD = 30 or 50. See: Cash Flow Column Descriptions, page 7-8.

4. The repriced rate defined in Step 3 equals the coupon rate that is used for amortization, prepayment, and interest income (financial element 430) calculations.

   Note: Note: If the Transfer Pricing rule has the Model with Gross Rates option selected, the cash flow engine uses the CUR_GROSS_RATE and MARGIN_GROSS for amortization and prepayment purposes.

5. MARGIN is also used in Oracle Transfer Pricing when mid-period repricing is selected for spread from Note Rate to compute the rate from a prior period.

Data Verification Requirements and Suggested Defaults

- For administered rate accounts, MARGIN = 0.
- For adjustable rate accounts with no contractual margin, MARGIN = 0.
- For applicable accounts, margin can be positive or negative.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Matched Spread (MATCHED_SPREAD)

Definition

Interest margin on a product, calculated by Oracle Transfer Pricing.

Module Usage

Oracle Transfer Pricing computes MATCHED_SPREAD when processing using the Standard calculation mode. It is calculated as follows:

For assets: Current Net Rate - Transfer Rate

For liabilities: Transfer Rate - Current Net Rate

Data Verification Requirements and Suggested Defaults

Since the cash flow engine calculates MATCHED_SPREAD, the default can be set to 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Maturity Date (MATURITY_DATE)

Definition

Contractual date on which the principal balance of an earning asset or debt instrument is due and payable to the holder.

Module Usage

MATURITY_DATE defines the final date of payment for a record. The MATURITY_DATE signals the end of processing for a given record.

1. MATURITY_DATE is referenced for fixed-rate straight term transfer pricing methodologies. When defining the transfer pricing term for a record, the cash flow engine subtracts the ORIGINATION_DATE from the MATURITY_DATE. The term is then matched to the relevant Interest Rate Code (IRC). The derived rate is applied to the record as the TRANSFER_RATE.

2. MATURITY_DATE is referenced by cash flow transfer pricing methodologies for both adjustable and fixed-rate records. For adjustable records, the cash flow engine transfer prices all cash flows on payment dates within the LAST_REPRICE_DATE and NEXT_REPRICE_DATE. The MATURITY_DATE is used to determine the last payment of a record.

3. The MATURITY_DATE is also referenced in order to determine the remaining number of payments for user-defined payment pattern records. The cash flow engine references the payment pattern payment frequencies and counts the number of payments from the ORIGINATION_DATE to the MATURITY_DATE.

4. Maturity Date is referenced during option cost processing for Fixed Rate and Adjustable rate instruments as cash flows are generated to maturity in both cases for these calculations.

Data Verification Requirements and Suggested Defaults

- For non-term accounts, MATURITY_DATE = CALENDAR_PERIOD + 1 Day.
- For term accounts, MATURITY_DATE is required.
- If the record is not past due or defaulted, MATURITY_DATE > CALENDAR_PERIOD.
- MATURITY_DATE = ORIGINATION_DATE + ORG_TERM.
- MATURITY_DATE = ISSUE_DATE + ISSUE_TERM.

- MATURITY_DATE = CALENDAR_PERIOD + REMAIN_TERM.
- MATURITY_DATE <= NEXT_PAYMENT_DATE + (REMAIN_NO_PMTS* PMT_FREQ).

  Assume that MATURITY_DATE is less than NEXT_PAYMENT_DATE + (REMAIN_NO_PMTS * PMT_FREQ). This implies that the calculated final payment date for an account differs from MATURITY_DATE (the condition results in what is called a stub payment). In this case, the cash flow engine forces the true last payment to be made on the maturity date. If this condition is not met (most likely caused because REMAIN_NO_PMTS is too low), the cash flow engine skips scheduled payments.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Negative Amortization Amount (NEG_AMRT_AMT)

Definition

The total amount of principal added to outstanding principal, resulting from payments which were not large enough to cover interest due.

Module Usage

The Oracle Transfer Pricing adjustable-type cash flow methodologies reference NEG_AMRT_AMT in calculating the current payment for negative amortization-type accounts. This is only relevant for adjustable-rate accounts with AMRT_TYPE_CODE = 600.

1. In a negatively amortizing record, the CUR_PAYMENT is less than the principal and interest that is due on the payment date. The interest portion that is not included in the payment goes to two places. It is added to the NEG_AMRT_AMT column and is added back to the principal amount. Because the NEG_AMRT_AMT balance is already included in the outstanding principal balance, NEG_AMRT_AMT is not explicitly used when the cash flow engine fully re-amortizes the account.

2. NEG_AMRT_AMT is used by the cash flow engine to keep track of negative amortization separately from non-negative amortization (normal) principal balance. It is separate for two reasons:

   - Because the cash flow engine pays down the negatively amortized portion before the principal portion, a separation of the two amounts must be done to allow the cash flow engine to identify what portion of the principal balance is

negatively amortized.

- When calculating the current payment, the cash flow engine uses NEG_AMRT_AMT in its check to see if NEG_AMRT_LIMIT has been exceeded. See: Negative Amortization Limit (NEG_AMRT_LIMIT), page 7-51.

Following is the process of events with regard to NEG_AMRT_AMT and related negative amortization fields:

1. Record is currently negatively amortizing since the payment amount, as defined by CUR_PAYMENT, is not enough to cover the principal and interest portion. The unpaid interest at each payment date goes into the NEG_AMRT_AMT field and back into the principal.

2. While calculating a payment event (payment date), if the engine calculates negative principal runoff, the cash flow engine also checks the negative amortization limit (NEG_AMRT_LIMIT) in order to ensure that the current NEG_AMRT_AMT is not exceeding its limit. NEG_AMRT_LIMIT is defined as a percentage of the original principal balance. If NEG_AMRT_AMT is exceeding this limit, the cash flow engine will recalculate the payment amount in order to fully amortize the instrument.

3. However, when deriving the recalculated payment amount after a NEG_AMRT_LIMIT has been exceeded, the cash flow engine also applies payment decrease/increase limits per period (PMT_DECR_CYCLE, PMT_INCR_CYCLE) and payment decrease/increase limits for the life of the record (PMT_DECR_LIFE, PMT_INCR_LIFE). Because these fields limit how much the CUR_PAYMENT can be changed, it is possible that the record will continue to negatively amortize even after a NEG_AMRT_LIMIT has been exceeded. If negative amortization does continue, the NEG_AMRT_AMT continues to grow.

4. The cash flow engine also attempts to recalculate the negatively amortizing payment amount on a PMT_ADJUST_DATE. Just like a payment recalculation for a NEG_AMRT_LIMIT, a payment recalculation on the PMT_ADJUST_DATE takes into account the effects of payment decrease/increase limits per period and payment decrease/increase limits for the life of the record. This allows for additional negative amortization to occur even after the PMT_ADJUST_DATE has recalculated the payment amount. PMT_ADJUST_DATE is incremented forward by the PMT_CHG_FREQ field until the maturity.

5. In addition to PMT_ADJUST_DATE and NEG_AMRT_LIMIT, the record can experience a payment recalculation on the negative amortization equalization date (NEG_AMRT_EQ_DATE). On this date, the CUR_PAYMENT for the record is fully re-amortized. NEG_AMRT_EQ_DATE ignores payment decrease/increase limits per period and payment decrease/increase limits for the life of the record. Therefore, after the payment recalculation of a NEG_AMRT_EQ_DATE, the record is no longer negatively amortizing and the NEG_AMRT_EQ_DATE is incremented forward by the NEG_AMRT_EQ_FREQ until maturity.

Data Verification Requirements and Suggested Defaults

- NEG_AMRT_LIMIT must be within the range of 0 to 100.
- For AMRT_TYPE_CODE <> 600, NEG_AMRT_AMT = 0.
- If AMRT_TYPE_CODE = 600, 0 <= NEG_AMRT_AMT <= NEG_AMRT_LIMIT/100 * ORG_PAR_BAL.
- If not applicable, default to 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Negative Amortization Equalization Date (NEG_AMRT_EQ_DATE)

Definition

The date on which a negative amortization-type account fully re-amortizes, regardless of payment caps and floors.

Module Usage

The Oracle Transfer Pricing adjustable-type cash flow methodologies reference NEG_AMRT_EQ_DATE when calculating the current payment for negative amortization-type accounts. NEG_AMRT_EQ_DATE is only relevant for adjustable-rate accounts with AMRT_TYPE_CODE = 600.

1. On the NEG_AMRT_EQ_DATE, the payment for a negatively amortizing record is recalculated. On this date, the CUR_PAYMENT for the record is fully re-amortized. NEG_AMRT_EQ_DATE ignores payment decrease/increase limits per period and payment decrease/increase limits for the life of the record. Therefore, after the payment recalculation of a NEG_AMRT_EQ_DATE, the record is no longer negatively amortizing.
2. NEG_AMRT_EQ_DATE is incremented forward by the NEG_AMRT_EQ_FREQ until the maturity date is reached.
3. For an explanation of NEG_AMRT_EQ_DATEs relationship with other related negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Note: If NEG_AMRT_EQ_FREQ = 0, once the modeling date is past

the NEG_AMRT_EQ_DATE, the cash flow engine does not attempt to re-amortize the negative amortized amount.

Data Verification Requirements and Suggested Defaults

- If AMRT_TYPE_CODE <> 600, NEG_AMRT_EQ_DATE = 01-Jan-19000.
- If AMRT_TYPE_CODE = 600, NEG_AMRT_EQ_DATE > ORIGINATION_DATE.
- If AMRT_TYPE_CODE = 600, NEG_AMRT_EQ_DATE < MATURITY_DATE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Negative Amortization Equalization Frequency (NEG_AMRT_EQ_FREQ)

Definition

Used in conjunction with NEG_AMRT_EQ_MULT to define the frequency with which negatively amortizing accounts are fully re-amortized.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference NEG_AMRT_EQ_FREQ in calculating the current payment for negative amortization-type accounts. This is only relevant for adjustable-rate accounts with AMRT_TYPE_CODE = 600.

1. From the NEG_AMRT_EQ_DATE, the cash flow engine rolls forward by the NEG_AMRT_EQ_FREQ until the maturity date.
2. At each instance of a NEG_AMRT_EQ_FREQ, the cash flow engine recalculates the payment as it did for the NEG_AMRT_EQ_DATE. On these roll dates, the CUR_PAYMENT for the record is fully re-amortized. NEG_AMRT_EQ_FREQ ignores payment decrease/increase limits per period and payment decrease/increase limits for the life of the record. Therefore, after the payment recalculation of a NEG_AMRT_EQ_FREQ, the record is no longer negatively amortizing.
3. If NEG_AMRT_EQ_FREQ = 0, once the modeling date is past the NEG_AMRT_EQ_DATE, the engine does not attempt to re-amortize the negative amortized amount. In this case, any negative amortized balance balloons at maturity.
4. For an explanation of NEG_AMRT_EQ_FREQs relationship with other related

negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- If AMRT_TYPE_CODE = 600, NEG_AMRT_EQ_FREQ must be either 0 or a positive value. NEG_AMRT_EQ_FREQ cannot be negative.
- If AMRT_TYPE_CODE <> 600, NEG_AMRT_EQ_FREQ = 0.
- If AMRT_TYPE_CODE = 600, 0 <= NEG_AMRT_EQ_FREQ < ORG_TERM.
- If AMRT_TYPE_CODE = 600, CALENDAR_PERIOD < NEG_AMRT_EQ_DATE <= MATURITY_DATE.
- Validation of NEG_AMRT_EQ_FREQ is always done in conjunction with NEG_AMRT_EQ_MULT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Negative Amortization Equalization Frequency Multiplier (NEG_AMRT_EQ_MULT)

Definition

Used in conjunction with NEG_AMRT_EQ_FREQ to define the frequency with which negatively amortizing accounts are fully re-amortized.

Module Usage

This column is the multiplier of the NEG_AMRT_EQ_FREQ column. It is used in conjunction with NEG_AMRT_EQ_FREQ to define the frequency with which negatively amortizing accounts are fully re-amortized. Oracle Transfer Pricing cash flow calculations reference NEG_AMRT_EQ_MULT when recalculating the current payment as defined in the NEG_AMRT_EQ_FREQ column. NEG_AMRT_EQ_MULT determines the units (Months, Days or Years) of NEG_AMRT_EQ_FREQ.

Data Verification Requirements and Suggested Defaults

- Valid values are:
  - D: Days
  - M: Months

- Y: Years

- Suggested default is M.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Negative Amortization Limit (NEG_AMRT_LIMIT)

Definition

Maximum negative amortization allowed as a percentage of the original balance.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference NEG_AMRT_LIMIT when determining if the NEG_AMRT_AMT is exceeding its defined limits. This is only relevant for adjustable-rate accounts with AMRT_TYPE_CODE = 600.

1. While calculating a payment event (payment date), if the cash flow engine calculates negative principal runoff, the cash flow engine also checks the negative amortization limit (NEG_AMRT_LIMIT) in order to ensure that the current NEG_AMRT_AMT is not exceeding its limit. NEG_AMRT_LIMIT is defined as a percentage of the original principal balance.

   For example, NEG_AMRT_LIMIT = 25 means that the negative amortization amount should never exceed 25% of the original principal balance (principal balance should never exceed 125% of the original balance). The formula for this check is:

   (-1 * calculated (negative) principal runoff + negative amortization balance > NEG_AMRT_LIMIT/100 * ORG_PAR_BAL)

   If NEG_AMRT_AMT is exceeding this limit, the cash flow engine recalculates the payment amount in order to fully amortize the instrument.

2. When deriving the recalculated payment amount after a NEG_AMRT_LIMIT has been reached, the cash flow engine also applies payment decrease/increase limits per period (PMT_DECR_CYCLE, PMT_INCR_CYCLE) and payment decrease/increase limits for the life of the record (PMT_DECR_LIFE, PMT_INCR_LIFE). Because these fields limit how much the CUR_PAYMENT can be changed, it is possible that the record continues to negatively amortize even after a NEG_AMRT_LIMIT has been exceeded. If negative amortization does continue, the NEG_AMRT_AMT continues to grow.

3. For an explanation of NEG_AMRT_LIMITs relationship with other related negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- If AMRT_TYPE_CODE <> 600, NEG_AMRT_LIMIT = 0.
- If AMRT_TYPE_CODE = 600, NEG_AMRT_LIMIT >= 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Net Margin Code (NET_MARGIN_CODE)

Definition

NET_MARGIN_CODE defines the relationship between CUR_GROSS_RATE and CUR_NET_RATE for the cash flow engine.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference NET_MARGIN_CODE when calculating a forecasted interest rate. NET_MARGIN_CODE has the following two, valid values:

- Floating Net Rate = 0
- Fixed Net Rate = 1

These are described below:

Floating Net Rate (NET_MARGIN_CODE = 0)

This is the default value for the Net Margin Code. If the detail data NET_MARGIN_CODE column is set to 0 (floating), the Existing Business record reprices at the relevant reprice dates as described under the NEXT_REPRICE_DATE section. Interest Income (FE 430) is based off of the CUR_NET_RATE.

Fixed Net Rate (NET_MARGIN_CODE = 1)

This setting is used by financial institutions that maintain the loans of other financial institutions. For example, Bank A may service (operate and process) the loans of Bank B. Bank B pays Bank A a fixed spread or margin as payment for maintaining the loans. Since Bank A receives a guaranteed fixed spread, only Bank B gains or loses when the actual loan reprices. For this reason, if the record reprices, Bank A should not

experience any change in interest income.

If the NET_MARGIN_CODE column of the detail record is set to 1 (fixed) and the Model With Gross Rates option is selected on the Transfer Pricing rule, the existing business record does not reprice even if the record is an adjustable-rate product (CUR_NET_RATE will not reprice). This is because it is assumed that the rate received by the bank (Bank A) equals the fixed spread that the bank is receiving as payment for maintaining the loans. The CUR_NET_RATE column of the record represents this fixed spread and is used for interest income (financial element 430) calculations while the CUR_GROSS_RATE for the record is used for prepayments and amortization.

> **Note:** The CUR_GROSS_RATE is used for amortization and prepayment calculation purposes and reflects the correct repriced rate for all such calculations.

If the NET_MARGIN_CODE is set to Fixed Net Rate, but the Model with Gross Rates option is not selected, the cash flow engine treats the records as if they are Floating Net Rate.

Data Verification Requirements and Suggested Defaults

- NET_MARGIN_CODE must be equal to 0 or 1.
- For fixed-rate accounts and adjustable accounts that reprice, NET_MARGIN_CODE = 0.
- For adjustable-rate accounts that represent records being serviced for a fixed fee, set NET_MARGIN_CODE = 1.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Next Payment Date (NEXT_PAYMENT_DATE)

Definition

Due date of the next scheduled payment. Otherwise known as cash flow date or date of runoff.

Module Usage

NEXT_PAYMENT_DATE is used to define the next scheduled (forecasted) payment date.

NEXT_PAYMENT_DATE is used for adjustable-rate cash flow transfer-priced records.

In defining the transfer rate for an adjustable rate record, the cash flow engine produces cash flows for each payment date from the LAST_REPRICE_DATE to the NEXT_REPRICE_DATE. In order to define all the payment events within this period, the cash flow engine rolls back from the NEXT_PAYMENT_DATE by the PMT_FREQ until just before the LAST_REPRICE_DATE. From this calculated payment date, the cash flow engine again rolls forward by the PMT_FREQ, but this time cash flows are produced. The cash flows produced are used by one of the three cash flow transfer pricing methodologies in order to derive the transfer rate.

If the remaining term calculation option is selected on the Transfer Pricing Process rule, the processing order in regards to NEXT_PAYMENT_DATE is as follows:

1. From the CALENDAR_PERIOD, the first cash flow event processed by the cash flow engine is the NEXT_PAYMENT_DATE. The cash flow engine references the NEXT_PAYMENT_DATE for the first forecasted payment date only. This applies to payment patterns (relative and absolute), but not for Payment Schedules.

2. From the NEXT_PAYMENT_DATE, the cash flow engine increments forward by the PMT_ FREQ until the MATURITY_DATE is reached.

   Note: For User-Defined Payment Schedules or Patterns, the cash flow engine does not reference the PMT_FREQ field. Instead, the cash flow engine references the schedule or pattern information to define the additional forecasted payment dates.

3. On the payment date the cash flow engine calculates the interest payments, principal payments, prepayments, and negative amortization, if applicable. Note that for an adjustable-type record where REPRICE_FREQ < PMT_FREQ, the cash flow engine applies only the last repriced rate for the purpose of payment calculation. If the record is not an AMRT_TYPE_CODE 700 (non-amortizing) or an AMRT_TYPE_CODE 600 (negatively amortizing), the principal balance of the record is reduced at each payment date.

   Note: The use of REMAIN_NO_PMTS varies depending on record type. For payment-patterned records, see:

   - Remaining Number of Payments (REMAIN_NO_PMTS), page 7-86.

   - Amortization Term (AMRT_TERM), page 7-16.

4. As each payment is made the cash flow engine reduces the REMAIN_NO_PMTS by 1. If the newly calculated REMAIN_NO_PMTS = 1, the next payment date is set to MATURITY_DATE.

Note: The use of REMAIN_NO_PMTS varies depending on record type. For special consideration for payment-patterned records, see:

- Remaining Number of Payments (REMAIN_NO_PMTS), page 7-86.
- Amortization Term (AMRT_TERM), page 7-16.

5. MATURITY_DATE is the final payment date. If the principal of a record is not reduced by the payment amounts, the remaining principal balance is paid on the MATURITY_DATE. For Payment Schedules, the cash flow engine does not use the NEXT_PAYMENT_ DATE columns. For these records, the cash flow engine makes the next payment on the first date in the schedule after the CALENDAR_PERIOD. However, for Payment Schedules and User-Defined Payment Patterns, the NEXT_PAYMENT_DATE from the detail record corresponds to the next defined payment date after the CALENDAR_PERIOD in the Schedule or Pattern interface.

Data Verification Requirements and Suggested Defaults

Required Conditions:

- NEXT_PAYMENT_DATE > CALENDAR_PERIOD.
- Also included in NEXT_PAYMENT_DATE is the modeling of past due, delinquent, or non-term accounts, if they are to be processed by the engine.

  For example, if the NEXT_PAYMENT_DATE is defaulted to 1900/01/01, the cash flow engine will roll the record by PMT_FREQ from that date until the maturity date. Note that this consumes considerable processing time.
- NEXT_PAYMENT_DATE <= MATURITY_DATE.
- If REMAIN_NO_PMTS > 1, NEXT_PAYMENT_DATE < MATURITY_DATE.
- If REMAIN_NO_PMTS = 1, NEXT_PAYMENT_DATE = MATURITY_DATE.
- MATURITY_DATE <= NEXT_PAYMENT_DATE + (REMAIN_NO_PMTS * PMT_FREQ).
- If the amortization type is an absolute pattern or payment schedule, the payment dates are predefined. The NEXT_PAYMENT_DATE on the detail record always corresponds to the relevant (next payment date after the CALENDAR_PERIOD) predefined payment date information of the Absolute Pattern interface or PAYMENT_SCHEDULE table.
- Suggested default:

If next payment date is unknown, set to either:

- CALENDAR_PERIOD + PMT_FREQ, or
- MATURITY_DATE

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Next Repricing Date (NEXT_REPRICE_DATE)

Definition

Date of next scheduled interest rate change for adjustable rate accounts.

Module Usage

NEXT_REPRICE_DATE defines the first forecasted repricing event after the Calendar Period date.

1. NEXT_REPRICE_DATE is used by adjustable-rate straight term transfer-priced records when using the Remaining Term calculation mode. The CALENDAR_PERIOD and NEXT_REPRICE_DATE define the term of the transfer pricing period. This term is matched to the relevant Interest Rate Code (IRC) to derive a transfer rate.

2. Adjustable-rate cash flow transfer-priced records use LAST_REPRICE_DATE and NEXT_REPRICE_DATE as the starting and ending points of the transfer-pricing period. In order to define all the payment events within this period, the cash flow engine rolls back from the NEXT_PAYMENT_DATE by the PMT_FREQ until just after the LAST_REPRICE_DATE. From this calculated payment date, the cash flow engine again rolls forward by the PMT_FREQ until just before the NEXT_REPRICE_DATE. As the cash flow engine rolls forward, cash flows are produced. The cash flows produced are used by one of the three cash flow transfer pricing methodologies in order to derive the transfer rate. See: Teaser-rate End Date (TEASER_END_DATE), page 7-90.

When calculating option costs, the following also applies when remaining term calculation mode is selected on the Transfer Pricing Process rule:

1. If the record is defined as ADJUSTABLE_TYPE_CODE = 250 and REPRICE_FREQ > 0, the cash flow engine references NEXT_REPRICE_DATE when calculating the first forecasted interest rate change.

Note: AMRT_TYPE_CODE definition does not impact whether the record is adjustable or not.

Note: If ADJUSTABLE_TYPE_CODE = 30 or 50, the cash flow engine does not reference the NEXT_ REPRICE_DATE. See: Adjustable Type Code (ADJUSTABLE_TYPE_CODE), page 7-8.

2. The cash flow engine rolls forward from NEXT_REPRICE_DATE to define the remaining forecasted Reprice Dates of the record. Rolling by the REPRICE_FREQ continues until MATURITY_DATE.

3. In defining the customer rate, on each reprice date, the cash flow engine matches the INTEREST_RATE_CODE, the reprice date and the REPRICE_FREQ of the record to the appropriate term point on the forecasted Interest Rate Code (IRC). To this derived rate, the cash flow engine adds the MARGIN (or MARGIN_GROSS, if applicable).

4. The cash flow engine then applies interest rate rounding and periodic/lifetime rate caps/floors. If the records TEASER_END_DATE is less than or equal to the CALENDAR_PERIOD, the cash flow engine applies the calculated forecasted rate to the record. Otherwise the cash flow engine applies the defined teased rate.

   Note: If multiple reprice dates exist within one payment period, (that is, if REPRICE_FREQ < PMT_FREQ) only the forecasted rate of the reprice date immediately preceding the payment dateis used for payment calculation purposes. The cash flow engine stores Before Reprice and After Reprice, Gross Rates and Net Rates as financial elements 260 to 290.

Data Verification Requirements and Suggested Defaults

- For fixed-rate accounts, NEXT_REPRICE_DATE = MATURITY_DATE.
- For administered rate accounts and floating rate accounts, use ADJUSTABLE_TYPE_CODE 30 or 50, which does not reference NEXT_REPRICE_DATE. Set the default to NEXT_REPRICE_DATE = NEXT_PAYMENT_DATE or MATURITY_DATE.
- If ADJUSTABLE_TYPE_CODE = 250 and repricing information is available, then:
  - NEXT_REPRICE_DATE > CALENDAR_PERIOD.
  - NEXT_REPRICE_DATE <= MATURITY_DATE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Original Payment Amount (ORG_PAYMENT_AMT)

Definition

The original payment amount at the date of origination.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference ORG_PAYMENT_AMT when referencing the payment amount at the time of the record origination.

1. Cash flow transfer pricing methodologies for fixed-rate records use ORG_PAYMENT_AMT as the payment amount for amortization purposes. For a fixed-rate record, the cash flow engine rolls forward from the ORIGINATION_DATE by PMT_FREQ when defining payment dates up to the MATURITY_DATE of the record. From origination, the cash flow engine amortizes the original balance (ORG_PAR_BAL) by the ORG_PAYMENT_AMT. The precise method of this amortization depends on the AMRT_TYPE_CODE.

2. ORG_PAYMENT_AMT is also used if the User-Defined Payment Pattern payment method % Original Payment is designated.

ORG_PAYMENT_AMT is also used in determining if the NEG_AMRT_AMT is exceeding its defined limits. This is only relevant for adjustable-rate accounts where AMRT_TYPE_CODE = 600.

1. For negative amortization-type accounts, the cash flow engine uses ORG_PAYMENT_AMT in determining whether a recalculated payment increase or decrease exceeds PMT_INCR_LIFE and PMT_DECR_LIFE.

2. On a recalculation date caused by a NEG_AMRT_LIMIT or PMT_ADJUST_DATE, the cash flow engine recalculates the payment amount in order to create a fully amortized record. After the recalculation, the cash flow engine references payment life increases/decreases columns (PMT_INCR_LIFE, PMT_DECR_LIFE). These fields limit the amount by which the recalculated payment amount can change from the original payment amount (ORG_PAYMENT_AMT).

   For example, if PMT_INCR_LIFE = 25.00, the recalculated payment amount is not allowed to increase by more than 25% of the ORG_PAYMENT_AMT. See:

   - Payment Increase Limit - Life (PMT_INCR_LIFE), page 7-73.

- Payment Decrease Limit - Life (PMT_DECR_LIFE), page 7-68.

Data Verification Requirements and Suggested Defaults

- If AMRT_TYPE_CODE <> 600, ORG_PAYMENT_AMT = 0.
- If adjustable-rate and AMRT_TYPE_CODE = 600, ORG_PAYMENT should be a valid non-zero value.
- ORG_PAYMENT may be validated using the following formula:
- ORG_PAYMENT = (ORG_BOOK_BAL* (CUR GROSS RATE/((12/PMT_FREQ[in months]) *100))) / (1 - ((1+(CUR_GROSS_RATE /((12/ PMT_FREQ [in months])*100))) ^-(ORG_TERM/PMT_FREQ [in months])))
- If ORG_PAYMENT_AMT is unknown, default to CUR_PAYMENT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Original Par Balance (ORG_PAR_BAL)

Definition

Represents the starting balance from which all fixed rate transfer pricing cash flows are generated, including principal run-off, prepayments, and interest cash flows.

Module Usage

Cash flow transfer pricing methodologies for fixed-rate records use ORG_PAR_BAL as the starting balance for all cash flow generation. For the treatment of adjustable-rate records, see: Last Reprice Date Balance (LRD_BALANCE), page 7-42.

The cash flow engine bases interest cash flows and principal runoff on ORG_PAR_BAL when transfer pricing cash flow methodology fixed-rate accounts. During processing, the cash flow engine rolls forward from the ORIGINATION_DATE by PMT_FREQ when defining payment dates up until the MATURITY_DATE of the record. From origination, the cash flow engine amortizes the original balance (ORG_PAR_BAL) by the ORG_PAYMENT_AMT. The precise method of this amortization depends on the AMRT_TYPE_CODE.

Data Verification Requirements and Suggested Defaults

- ORG_PAR_BAL requires a valid balance for all accounts.

- ORG_PAR_BAL = ORG_BOOK_BAL - DEFERRED_ORG_BAL .
- ORG_PAR_BAL must have the same sign as ORG_PAYMENT_AMT and CUR_PAYMENT.

  For the transfer pricing of fixed-rate instruments, the original balance should be populated. If REPRICE_FREQ = 0, ORG_PAR_BAL <> 0.
- Original balance on Rule of 78 instruments should not be greater than the current balance. If AMRT_TYPE_CODE = 710, ORG_PAR_BAL should be less than CUR_PAR_BAL.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Original Term (ORG_TERM)

Definition

Used in conjunction with ORG_TERM_MULT to define the contractual term at origination date.

Module Usage

The ORG_TERM of the instrument is referenced by Oracle Transfer Pricing as the period from ORIGINATION_DATE to MATURITY_DATE.

ORG_TERM is referenced by Oracle Transfer Pricing when calculating cash flows for fixed-rate cash flow methodologies.

1. Oracle Transfer Pricing cash flow methodologies use ORG_TERM when calculating the current payment for adjustable-rate accounts and in transfer pricing fixed-rate accounts. For adjustable-rate accounts, the cash flow engine compares ORG_TERM against AMRT_TERM, checking to see if the account is a balloon-type account.
2. If ORG_TERM < AMRT_TERM, the engine recognizes the record as a balloon. The cash flow engine amortizes the outstanding principal balance over the calculated number of payments, based upon the amortization maturity date. See: Amortization Term (AMRT_TERM), page 7-16.

Data Verification Requirements and Suggested Defaults

- ORG_TERM > 0.
- ORG_TERM <= ISSUE_TERM.

- ORG_TERM <= AMRT_TERM.
- ORG_TERM >= REPRICE_FREQ.
- ORG_TERM >= PMT_FREQ.
- ORG_TERM >= REMAIN_TERM.
- ORG_TERM + ORIGINATION_DATE = MATURITY_DATE.

Note: Validation of ORG_TERM should always be done in conjunction with ORG_TERM_MULT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Original Term Multiplier (ORG_TERM_MULT)

Definition

Used in conjunction with ORG_TERM to define the contractual term at origination date.

Module Usage

ORG_TERM_MULT is referenced by Oracle Transfer Pricing when calculating cash flows for fixed-rate cash flow methodologies.

Data Verification Requirements and Suggested Defaults

- Valid values are:
  - D: Days
  - M: Months
  - Y: Years
- For non-term accounts, default to M.
- For information on ORG_TERM_MULT validation, see: Original Term (ORG_TERM), page 7-60.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Origination Date (ORIGINATION_DATE)

Definition

The date the current financial institution originated or acquired the instrument.

Module Usage

Oracle Transfer Pricing references the ORIGINATION_DATE as the start date of the record. For additional information, see: Issue Date (ISSUE_DATE), page 7-39.

Prepayment assumptions also reference ORIGINATION_DATE. Separate assumptions can be defined for ORIGINATION_DATE ranges.

Additionally, when using a Prepayment Table rule:

- If the prepayment rate is driven by the Expired Term, the ORIGINATION_DATE is used to determine the age of the instrument using the following formula:

  (ROUND(Current Bucket Date - ORIGINATION_DATE)/30.42, 0)

- If the prepayment rate is driven by the ORIGINATION_DATE and the instrument is still in its tease period (i.e., TEASE_END_DATE > Current Bucket Date), then the REPRICE_FREQ is calculated as:

  (ROUND(TEASE_END_DATE - ORIGINATION_DATE)/30.42,0)

1. Fixed-rate cash flow transfer-priced records reference ORIGINATION_DATE in order to calculate the payment dates for amortization purposes.

   Note: For adjustable rate accounts, transfer pricing cash flow calculations begin on LAST_REPRICE_DATE.

   Note: When defining the payment dates for the record, the cash flow engine starts from the records ORIGINATION_DATE and rolls forward by PMT_FREQ until the MATURITY_DATE is reached. The cash flow engine bases interest cash flows and principal runoff on ORG_PAR_BAL when transfer pricing cash flow methodology fixed-rate accounts. From ORIGINATION_DATE, the cash flow engine amortizes the original balance (ORG_PAR_BAL) by the ORG_PAYMENT_AMT. The

precise method of this amortization depends on the AMRT_TYPE_CODE.

2. Straight term methodology references ORIGINATION_DATE when defining the transfer pricing term that is matched to the term on the yield curve (Transfer Pricing Interest Rate Code). For fixed-rate instruments, the term defined by MATURITY_DATE - ORIGINATION_DATE is matched to the relevant Interest Rate Code (IRC).

   For adjustable-rate instruments in their tease period, the term is figured as the (TEASE_END_DATE - ORIGINATION_DATE). The transfer pricing assignment date for the IRC is also determined by the ORIGINATION_DATE of the record. That is, the date of the yield curve (IRC) is matched to the date of the record origination.

3. If the record is transfer-priced using a Spread From Interest Rate Code or Redemption Curve methodology, the option of choosing the IRCs assignment date is available. If the Origination Date is chosen as the assignment date, or if the assignment date is the Last Repricing Date and the instrument is fixed rate, the date of the IRC used for transfer rate calculations is the same as the detail records ORIGINATION_DATE. If an IRC of the same date does not exist the cash flow engine uses yield curve information for the closest preceding date.

4. For records that reference the User-Defined Payment Patterns, the cash flow engine derives the remaining number of payments by counting the number of payments from the ORIGINATION_DATE to the MATURITY_DATE.

Data Verification Requirements and Suggested Defaults

- For all accounts, a valid ORIGINATION_DATE is required.
- ORIGINATION_DATE >= ISSUE_DATE.
- ORIGINATION_DATE <= CALENDAR_PERIOD (unless future originations booked on system).
- ORIGINATION_DATE <= LAST_REPRICE_DATE.
- For term accounts only:
  - ORIGINATION_DATE + ORG_TERM = MATURITY_DATE.
  - ORIGINATION_DATE <= TEASER_END_DATE (if TEASER_END_DATE is valid).

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Adjustment Date (PMT_ADJUST_DATE)

Definition

Date of next payment adjustment for adjustable-rate, negative amortization-type accounts.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_ADJUST_DATE when processing and calculating the current payment for negative amortization-type accounts. This is only relevant for adjustable-rate accounts where AMRT_TYPE_CODE = 600.

1. The cash flow engine recalculates negatively amortizing payment amounts on a PMT_ADJUST_DATE. Just like a payment recalculation for a NEG_AMRT_LIMIT, a payment recalculation on the PMT_ADJUST_DATE takes into account the effects of payment decrease/increase limits per period (PMT_DECR_CYCLE, PMT_INCR_CYCLE) and payment decrease/increase limits for the life of the record (PMT_DECR_LIFE, PMT_INCR_LIFE). This allows for additional negative amortization to occur even after the PMT_ADJUST_DATE has recalculated the payment amount.

2. PMT_ADJUST_DATE is incremented forward by the PMT_CHG_FREQ field until maturity.

3. PMT_ADJUST_DATE differs from NEG_AMRT_EQ_DATE because on PMT_ADJUST_DATE, the calculated payment is constrained by payment decrease/increase limits per period and payment decrease/increase limits for the life of the record. However, on the NEG_AMRT_EQ_DATE, the calculated payment overrides these payment change limits.

4. For an explanation of the relationships among the relevant negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- For fixed-rate and non-term accounts, PMT_ADJUST_DATE = 01-Jan-19000.

- For adjustable-rate accounts with AMRT_TYPE_CODE <> 600, PMT_ADJUST_DATE = NEXT_REPRICE_DATE.

- For adjustable-rate accounts with AMRT_TYPE_CODE = 600, the following conditions should exist:
  - PMT_ADJUST_DATE > CALENDAR_PERIOD.
  - PMT_ADJUST_DATE > ORIGINATION_DATE.
  - PMT_ADJUST_DATE <= CALENDAR_PERIOD + PMT_CHG_FREQ) PMT_ADJUST_DATE <= MATURITY_DATE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Change Frequency (PMT_CHG_FREQ)

Definition

Used in conjunction with PMT_CHG_FREQ_MULT to define the frequency at which the payment for an account adjusts.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_CHG_FREQ when processing and calculating the current payment for negative amortization-type accounts. This is only relevant for adjustable-rate accounts where AMRT_TYPE_CODE = 600.

1. The PMT_CHG_FREQ is used to increment forward from the PMT_ADJUST_DATE.
2. The cash flow engine recalculates negatively amortizing payment amounts on a PMT_CHG_FREQ. Just like a payment recalculation for a NEG_AMRT_LIMIT, a payment recalculation on the PMT_CHG_FREQ takes into account the effects of payment decrease/increase limits per period (PMT_DECR_CYCLE, PMT_INCR_CYCLE) and payment decrease/increase limits for the life of the record (PMT_DECR_LIFE, PMT_INCR_LIFE). This allows for additional negative amortization to occur even after the PMT_CHG_FREQ event has recalculated the payment amount.
3. Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- If AMRT_TYPE_CODE = 600, PMT_CHG_FREQ must be either 0 or a positive

value. PMT_CHG_FREQ cannot be negative.

- For fixed-rate and non-term accounts, PMT_CHG_FREQ = 0.
- For adjustable-rate accounts with AMRT_TYPE_CODE <> 600, PMT_CHG_FREQ = REPRICE_FREQ.
- For adjustable-rate accounts with AMRT_TYPE_CODE = 600, the following conditions should exist:
  - PMT_CHG_FREQ <> 0.
  - PMT_CHG_FREQ <= PMT_ADJUST_DATE - CALENDAR_PERIOD.

Note: Validation of PMT_CHG_FREQ should always be done in conjunction with PMT_CHG_FREQ_MULT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Change Frequency Multiplier (PMT_CHG_FREQ_MULT)

Definition

Used in conjunction with PMT_CHG_FREQ to define the frequency at which the payment for an account adjusts.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_CHG_FREQ_MULT when processing and calculating the current payment for negative amortization-type accounts. PMT_CHG_FREQ_MULT determines the units (months, days or years) of PMT_CHG_FREQ.

Data Verification Requirements and Suggested Defaults

- Valid values are:
  - D: Days
  - M: Months
  - Y: Years

- For information on PMT_CHG_FREQ_MULT validation, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Decrease Limit - Cycle (PMT_DECR_CYCLE)

Definition

Maximum payment decrease allowed during the payment change cycle of an adjustable-rate instrument.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_DECR_CYCLE when recalculating the current payment for negative amortization events such as NEG_AMRT_LIMIT, PMT_ADJUST_DATE, and PMT_CHG_FREQ. This is only relevant for adjustable-rate accounts where AMRT_TYPE_CODE = 600.

1. For negative amortization-type accounts, the cash flow engine uses PMT_DECR_CYCLE to calculate the maximum decrease in the payment amount allowed from the previous payment change to the next.

2. PMT_DECR_CYCLE is defined in terms of a percentage. The engine performs the following check:

   Previous Current Payment - Newly calculated payment > (PMT_DECR_CYCLE/100 * Previous Current Payment)

   If the newly calculated payment satisfies the above equation, the engine limits the decrease to the amount = (PMT_DECR_CYCLE * Previous Current Payment).

   For example, if PMT_DECR_CYCLE = 5.00, the calculated current payment is not allowed to decrease by more than 5% of the previous current payment.

3. If PMT_DECR_CYCLE = 0, the cash flow engine assumes that there is no payment decrease limit per payment change period.

4. The PMT_DECR_CYCLE is referenced when the following negative amortization events occur: NEG_AMRT_LIMIT, PMT_ADJUST_DATE, and PMT_CHG_FREQ.

5. For an explanation of the relationship of the PMT_DECR_CYCLE with other related negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- For accounts with AMRT_TYPE_CODE <> 600, PMT_DECR_CYCLE = 0.
- For accounts with AMRT_TYPE_CODE = 600, 0 <= PMT_DECR_CYCLE < 100.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Decrease Limit - Life (PMT_DECR_LIFE)

Definition

Maximum payment decrease allowed during the life of an adjustable-rate instrument.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_DECR_LIFE when recalculating the current payment for negative amortization events such as NEG_AMRT_LIMIT, PMT_ADJUST_DATE, and PMT_CHG_FREQ. This is only relevant for adjustable-rate accounts where AMRT_TYPE_CODE = 600.

1. For negative amortization-type accounts, the cash flow engine uses PMT_DECR_LIFE to calculate the maximum decrease in the payment allowed during the life of the account. PMT_DECR_LIFE is defined in terms of a percentage of ORG_PAYMENT. The engine performs the following check:

   ORG_PAYMENT - Newly calculated payment > (PMT_DECR_LIFE/100 * ORG_PAYMENT)

   If the newly calculated payment satisfies the above equation, the cash flow engine limits the decrease to the amount = (PMT_DECR_LIFE * ORG_PAYMENT).

   For example, if PMT_DECR_LIFE = 25.00, the calculated current payment is not allowed to decrease by more than 25% of ORG_PAYMENT.

2. If PMT_DECR_LIFE = 0, the cash flow engine assumes that there is no lifetime payment decrease limit.

3. The PMT_DECR_LIFE field is referenced when the following negative amortization events occur:

   - NEG_AMRT_LIMIT
   - PMT_ADJUST_DATE

- PMT_CHG_FREQ

4. For an explanation of the relationship of PMT_DECR_LIFE with other related negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- For accounts with AMRT_TYPE_CODE <> 600, PMT_DECR_LIFE = 0.
- For accounts with AMRT_TYPE_CODE = 600, 0 <= PMT_DECR_LIFE < 100.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Frequency (PMT_FREQ)

Definition

Used in conjunction with PMT_FREQ_MULT to define the payment frequency of an account.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_FREQ for calculating and processing payments and prepayments. The engine forecasts future next payment dates by incrementing NEXT_PAYMENT_DATE forward by PMT_FREQ.

PMT_FREQ is referenced by the cash flow transfer pricing methodologies when deriving discounted cash flows. See: Detail Cash Flow Data, page 6-35.

Oracle Transfer Pricing uses PMT_FREQ in defining payment dates.

1. Oracle Transfer Pricing defines the date of payment using PMT_FREQ in the following ways:
   - **Adjustable-rate Cash Flow Transfer-priced Records:** In defining the transfer rate for an adjustable-rate record, the cash flow engine uses the PMT_FREQ to define the payment dates from the LAST_REPRICE_DATE to the NEXT_REPRICE_DATE. In order to define all the payment events within this period, the cash flow engine rolls back from the NEXT_PAYMENT_DATE by the PMT_FREQ until just before the LAST_REPRICE_DATE. From this calculated payment date, the cash flow engine again rolls forward by the PMT_FREQ, but this time cash flows are produced. The cash flows produced are used by one of the three cash flow transfer pricing methodologies in order

to derive the transfer rate.

- **Fixed-rate Cash Flow Transfer-priced Records:** In defining the payment dates for cash flow transfer-priced fixed-rate records, the cash flow engine starts from the ORIGINATION_DATE of the record and rolls forward by the PMT_FREQ until the MATURITY_DATE.

  Note: If the transfer pricing record utilizes User-Defined Payment Schedules or Patterns, PMT_FREQ is not referenced for payment date information.

2. Oracle Transfer Pricing utilizes the PMT_FREQ in the following manner:

   - On date of payment, the cash flow engine calculates the interest payments, principal payments, current deferred payments, prepayments, unscheduled prepayments, and negative amortization, if applicable.

   - During the payment calculation processing, PMT_FREQ is used for interest income calculations (financial element 430) for records with ACCRUAL_BASIS_CODEs of 30/360, 30/365, 30/ACTUAL. See:

     - Accrual Basis Code (ACCRUAL_BASIS_CODE), page 7-6.

     - Interest Type Code (INT_TYPE_CODE), page 7-37.

   - PMT_FREQ is used in the Remaining Number of Payments calculation when calculating the payment amounts for balloon records and specific user-defined payment pattern instances. See: Amortization Term (AMRT_TERM), page 7-16.

   - As each payment is made on the PMT_FREQ, the cash flow engine reduces the REMAIN_NO_PMTS by 1. If the newly calculated REMAIN_NO_PMTS = 1, the next payment date is set to MATURITY_DATE.

     Note: The use of REMAIN_NO_PMTS varies depending on record type. See:

     - Remaining Number of Payments (REMAIN_NO_PMTS), page 7-86.

     - Amortization Term (AMRT_TERM), page 7-16.

Data Verification Requirements and Suggested Defaults

- PMT_FREQ > 0 in all cases. PMT_FREQ = 0 can cause incorrect calculations to

occur.

- PMT_FREQ <= ORG_TERM
- MATURITY_DATE <= NEXT_PAYMENT_DATE + (REMAIN_NO_PMTS * PMT_FREQ).

  Note that PMT_FREQ = 1 and PMT_FREQ MULT = D requires the cash flow engine to perform cash flow calculations for every day of the modeling horizon. This slows processing significantly.

  Note: Validation of PMT_FREQ should always be done in conjunction with PMT_FREQ_MULT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Frequency Multiplier (PMT_FREQ_MULT)

Definition

Used in conjunction with PMT_FREQ to define the payment frequency of an account.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_FREQ_MULT for calculating and processing payments. PMT_FREQ_MULT determines the units (months, days, or years) of PMT_FREQ.

Data Verification Requirements and Suggested Defaults

- Valid values are:
  - D: Days
  - M: Months
  - Y: Years
- For information on PMT_FREQ_MULT validation, see: Payment Frequency (PMT_FREQ), page 7-69.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Increase Limit - Cycle (PMT_INCR_CYCLE)

Definition

Maximum payment increase allowed during the payment change cycle of an adjust-able-rate instrument.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_INCR_CYCLE when recalculating the current payment for negative amortization events such as NEG_AMRT_LIMIT, PMT_ADJUST_DATE, and PMT_CHG_FREQ. This is only relevant for adjustable-rate accounts where AMRT_TYPE_CODE = 600.

1. For negative amortization-type accounts, the cash flow engine uses PMT_INCR_CYCLE to calculate the maximum increase in the payment amount allowed from the previous payment change to the next.

2. PMT_INCR_CYCLE is defined in terms of a percentage. The cash flow engine performs the following check:

   Newly Calculated Payment - Previous Current Payment > (PMT_INCR_CYCLE/100 * Previous Current Payment)

   If the newly calculated payment satisfies the above equation, the cash flow engine limits the increase to the amount = (PMT_INCR_CYCLE * Previous Current Payment).

   For example, if PMT_INCR_CYCLE = 5.00, the calculated current payment is not allowed to increase by more than 5% of the previous current payment.

3. If PMT_INCR_CYCLE = 0, the cash flow engine assumes that there is no payment increase limit per payment change period.

4. The PMT_INCR_CYCLE is referenced when the following negative amortization events occur:

   - NEG_AMRT_LIMIT
   - PMT_ADJUST_DATE
   - PMT_CHG_FREQ

5. For an explanation of PMT_INCR_CYCLEs relationship with other related negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- If AMRT_TYPE_CODE <> 600, PMT_INCR_CYCLE = 0.
- If AMRT_TYPE_CODE = 600, 0 <= PMT_INCR_CYCLE < 100.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Payment Increase Limit - Life (PMT_INCR_LIFE)

Definition

Maximum payment increase allowed during the life of an adjustable-rate instrument.

Module Usage

Oracle Transfer Pricing cash flow methodologies reference PMT_INCR_LIFE when recalculating the current payment for negative amortization events such as NEG_AMRT_LIMIT, PMT_ADJUST_DATE, and PMT_CHG_FREQ. This is only relevant for adjustable-rate accounts where AMRT_TYPE_CODE = 600.

1. For negative amortization-type accounts, the cash flow engine uses PMT_INCR_LIFE to calculate the maximum increase in the payment allowed during the life of the account.

2. PMT_INCR_LIFE is defined in terms of a percentage of ORG_PAYMENT. The cash flow engine performs the following check:

   Newly calculated payment - ORG_PAYMENT > (PMT_INCR_LIFE/100 * ORG_PAYMENT)

   If the newly calculated payment satisfies the above equation, the cash flow engine limits the increase to the amount = (PMT_INCR_LIFE * ORG_PAYMENT).

   For example, if PMT_INCR_LIFE = 25.00, the calculated current payment is not allowed to increase by more than 25% of ORG_PAYMENT.

3. The PMT_INCR_LIFE is referenced when the following negative amortization events occur:

   - NEG_AMRT_LIMIT

- PMT_ADJUST_DATE
- PMT_CHG_FREQ

4. For an explanation of PMT_INCR_LIFEs relationship with other related negative amortization fields, see: Negative Amortization Amount (NEG_AMRT_AMT), page 7-46.

Data Verification Requirements and Suggested Defaults

- If PMT_INCR_LIFE = 0, the cash flow engine assumes that there is no lifetime payment increase limit.
- If AMRT_TYPE_CODE <> 600, PMT_INCR_LIFE = 0
- If AMRT_TYPE_CODE = 600, 0 <= PMT_INCR_LIFE < 100

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Percent Sold (PERCENT_SOLD)

Definition

Percent of balance sold to investors.

Module Usage

PERCENT_SOLD is used to calculate net balance and net payment when an instrument is partially participated out to another financial institution. In the case of a participated loan, the bank partially owns the loan. A bank participates out the balance of the loan to another financial institution if it is not able to lend the entire amount or if lending the entire amount exceeds the legal lending limits of the bank. By selling most of the financing to another financial institution, the bank earns fee income from servicing the loan and is able to retain other banking relationships, such as checking accounts.

1. Oracle Transfer Pricing cash flow methodologies use PERCENT_SOLD to perform all net balance calculations.

2. The cash flow engine performs all of the cash flow calculations on a gross balance basis, but must net out the principal and interest portions not owned/owed by the bank. PERCENT_SOLD defines the percent of the balances participated (sold) by the bank.

3. The calculation performed to net out participations sold is:

   [Net Balance] = [Gross Balance] * (100 - PERCENT_SOLD)/100

4. PERCENT_SOLD does not apply if an account has been sold to another subsidiary of the same company. In the case of accounts that have had portions sold from one legal entity of a holding company to another, PERCENT_SOLD = 0.

5. For wholly (100%) owned accounts, PERCENT_SOLD = 0.

Data Verification Requirements and Suggested Defaults

For accounts with PERCENT_SOLD <> 0, the following conditions should exist:

- 0 < PERCENT_SOLD <= 100.
- CUR_NET_BOOK_BAL = CUR_BOOK_BAL * (100 - PERCENT_SOLD) / 100.
- CUR_NET_PAR_BAL = CUR_PAR_BAL * (100 - PERCENT_SOLD) / 100.
- ORG_NET_BOOK_BAL = ORG_BOOK_BAL * (100 - PERCENT_SOLD) / 100.
- ORG_NET_PAR_BAL = ORG_PAR_BAL * (100 - PERCENT_SOLD) / 100.
- If not applicable, PERCENT_SOLD = 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Prior Transfer Pricing Period Average Daily Balance (PRIOR_TP_PER_ADB)

Definition

The average balance at the reprice date prior to the LAST_REPRICE_DATE.

Module Usage

When processing with the mid-period repricing option, Oracle Transfer Pricing references PRIOR_TP_PER_ADB as the average daily balance at the time of the last repricing event prior to the LAST_REPRICE_DATE. This column is used in conjunction with the CUR_TP_PER_ADB field.

1. Mid-period repricing produces an average transfer rate over the current processing month if the LAST_REPRICE_DATE occurred since the beginning of the processing month. PRIOR_TP_PER_ADB and CUR_TP_PER_ADB are used as average balance

weightings in the mid-period repricing equation. PRIOR_TP_PER_ADB is used to determine the balance on the reprice date prior to the LAST_REPRICE_DATE and CUR_TP_PER_ADB is used to determine the balance as of the LAST_REPRICE_DATE.

For an example and explanation of the relationship between PRIOR_TP_PER_ADB and CUR_TP_PER_ADB, see: Current Transfer Pricing Period Average Daily Balance (CUR_TP_PER_ADB), page 7-29.

2. If the CUR_TP_PER_ADB and PRIOR_TP_PER_ADB are not available, use CUR_PAR_BAL as the default.

3. If the TEASER_END_DATE is greater than the CALENDAR_PERIOD, the mid-period repricing does not apply and the CUR_TP_PER_ADB and PRIOR_TP_PER_ADB columns are not used.

Data Verification Requirements and Suggested Defaults

- If the record is adjustable and repricing occurs within the month, PRIOR_TP_PER_ADB = (average) balance at the time of the reprice date prior to the LAST_REPRICE_DATE.

- If the average balance at the time of the reprice date prior to the LAST_REPRICE_DATE is not available, use CUR_PAR_BAL.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Cap Life (RATE_CAP_LIFE)

Definition

Maximum interest rate allowed during life of an adjustable-rate instrument.

Module Usage

Oracle Transfer Pricing references RATE_CAP_LIFE when calculating option costs.

For adjustable-rate records, the cash flow engine matches the REPRICE_FREQ, INTEREST_RATE_CODE, and the reprice date to the information generated by the Monte Carlo rate engine. The margin is then added to this forecasted rate. Any rounding, rate caps (RATE_CAP_LIFE) or floors, and tease period are applied, and the resulting rate is applied to the record as its repriced rate.

Data Verification Requirements and Suggested Defaults

- If the record is fixed-rate, RATE_CAP_LIFE = 0.
- If the record is adjustable without a rate cap, RATE_CAP_LIFE = 0.
- If RATE_CAP_LIFE <> 0, CUR_GROSS_RATE and CUR_NET_RATE <= RATE_CAP_LIFE.
- If RATE_CAP_LIFE <> 0, RATE_CAP_LIFE = ORG_RATE + RATE_INCR_LIFE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Change Minimum (RATE_CHG_MIN)

Definition

Minimum change in the repricing index that is necessary for a change to be made to the interest rate.

Module Usage

Oracle Transfer Pricing references RATE_CHG_MIN during option cost processing when calculating a forecasted interest rate.

1. For adjustable-rate records, the cash flow engine matches the REPRICE_ FREQ, INTEREST_RATE_CODE, and the reprice date to the information generated by the Monte Carlo rate engine. The margin is then added to this forecasted rate. Any rounding is applied, followed by a check of the RATE_CHG_MIN, when determining the fully indexed rate. Rate caps or floors and tease periods are applied, and the resulting rate is used as the repriced rate for the record.

   Note: ADJUSTABLE_TYPE_CODE = 30 or 50 does not reference reprice dates.

2. If the absolute value of [(forecasted rate + margin) - (previous rate + margin)] < RATE_CHG_MIN, the cash flow engine sets the new forecasted rate = previous rate. The cash flow engine does not change the previous rate to the new forecasted rate. The previous rate is defined as either the rate on the detail record (CUR_ NET_RATE or CUR_GROSS_RATE) or the previous forecasted rate from the Monte Carlo engine. After the RATE_CHG_MIN is calculated any other cap/floor, rounding, and tease periods is then applied. See:

- Next Repricing Date (NEXT_REPRICE_DATE), page 7-56.
- Overview of Cash Flow Calculations, page 6-1.

Data Verification Requirements and Suggested Defaults

- If the record is fixed-rate, RATE_CHG_MIN = 0.
- If the record is adjustable, without a minimum rate change amount, RATE_CHG_MIN = 0.
- If RATE_CHG_MIN <> 0, 0 < RATE_CHG_MIN <= 1.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Change Rounding Code (RATE_CHG_RND_CODE)

Definition

Method used for rounding of the interest rate change.

Module Usage

Oracle Transfer Pricing references RATE_CHG_RND_CODE when calculating option costs.

The cash flow engine uses RATE_CHG_RND_CODE to determine the rounding method that is applied to the current rate after a repricing event. RATE_CHG_ RND_CODE is used in conjunction with RATE_CHG_RND_FAC.

1. For adjustable-rate records, the cash flow engine matches the REPRICE_ FREQ, INTEREST_RATE_CODE, and the reprice date to the information generated by the Monte Carlo rate engine. The margin is then added to this forecasted rate. Any rounding (RATE_CHG_RND_CODE and RATE_ CHG_RND_FAC), rate caps/floors, and tease periods are applied, and the resulting rate is applied to the record as its repriced rate.

   Note: ADJUSTABLE_TYPE_CODE = 30 or 50 does not reference reprice dates.

2. The RATE_CHG_RND_CODE accepts values 1 to 4. Depending on the value input, the value of the forecasted rate differs. The following table describes the possible

values of the RATE_CHG_RND_CODE:

Rate Change Rounding Codes

Rate Change Rounding Code	Description
RATE_CHG_RND_CD = 1	**Truncate:** The cash flow engine truncates the forecasted rate to a whole value. For example, if unrounded forecast = 8.65, the truncated forecasted rate = 8.00.
RATE_CHG_RND_CD = 2	**Round Up:** The engine rounds the rate up the nearest multiple of the RATE_CHG_RND_FAC. For example, if the unrounded forecasted rate is 8.65 and the RATE_CHG_RND_FAC = 0.5, the rounded forecasted rate = 9.00.
RATE_CHG_RND_CD = 3	**Round Down:** The engine rounds the rate down to the nearest multiple of RATE_CHG_ RND_FAC. For example, if the unrounded forecasted rate is 8.65 and the RATE_CHG_ RND_FAC = 0.25, the rounded forecasted rate = 8.50.
RATE_CHG_RND_CD = 4	**Round Nearest:** The engine rounds the rate to the nearest multiple of RATE_CHG_RND_FAC. For example, if the unrounded forecasted rate is 8.65 and the RATE_CHG_RND_FAC = 0.25, the rounded forecasted rate = 8.75.

Data Verification Requirements and Suggested Defaults

- RATE_CHG_RND_CODE must be within the range 0 to 4.
- For fixed-rate accounts, RATE_CHG_RND_CODE = 4 and RATE_CHG_RND_FAC = 0.
- For variable-rate accounts with rates that are not rounded, RATE_CHG_RND_CODE = 4 and RATE_CHG_RND_FAC = 0.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Change Rounding Factor (RATE_CHG_RND_FAC)

Definition

Factor by which the rate change on an adjustable instrument is rounded.

Module Usage

Oracle Transfer Pricing references RATE_CHG_RND_FAC when calculating option costs.

The cash flow engine references RATE_CHG_RND_FAC when calculating a forecasted interest rate, and contains the value to which forecasted interest rates are rounded. RATE_CHG_RND_FAC is used in conjunction with RATE_CHG_RND_ CODE.

1. For adjustable-rate records, the cash flow engine will match the REPRICE_ FREQ, INTEREST_RATE_CODE, and the reprice date to the information generated by the monte carlo rate engine. The margin will then be added to this forecasted rate. Any rounding (RATE_CHG_RND_CODE and RATE_ CHG_RND_FAC), rate caps/floors, and tease periods will be applied, and the resulting rate is applied to the record as the records repriced rate.

   Note: ADJUSTABLE_TYPE_CODE = 30 or 50 does not reference reprice dates.

2. For an explanation of usage and the relationship between RATE_CHG_RND_ FAC and RATE_CHG_RND_CODE, see: Rate Change Rounding Code (RATE_CHG_RND_CODE), page 7-78.

Data Verification Requirements and Suggested Defaults

- For fixed-rate accounts, RATE_CHG_RND_CODE = 4 and RATE_CHG_RND_FAC = 0.
- For variable-rate accounts with rates that are not rounded, RATE_CHG_RND_CODE = 4 and RATE_CHG_RND_FAC = 0.
- For variable-rate accounts with RATE_CHG_RND_FAC <> 0, 0 < RATE_CHG_RND_FAC <= 1.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Decrease Limit - Cycle (RATE_DECR_CYCLE)

Definition

Maximum rate decrease allowed during a repricing cycle for an adjustable-rate instrument.

Module Usage

Oracle Transfer Pricing references RATE_DECR_CYCLE when calculating option costs.

The cash flow engine references RATE_DECR_CYCLE when calculating a forecasted interest rate. RATE_DECR_CYCLE sets the maximum amount (in terms of basis points) by which the interest rate may decrease in a given REPRICE_FREQ.

1. For adjustable-rate records, the cash flow engine matches the REPRICE_ FREQ, INTEREST_RATE_CODE, and the reprice date to the information generated by the Monte Carlo rate engine. The margin is then added to this forecasted rate. Any rounding, rate caps/floors (RATE_DECR_ CYCLE), and tease periods are applied, and the resulting rate is applied to the record as its repriced rate.

   Note: ADJUSTABLE_TYPE_CODE = 30 or 50 does not reference reprice dates.

2. When applying the RATE_DECR_CYCLE, the engine checks for the following:

   - Previous Current Rate > Calculated forecasted rate
   - Previous Current Rate - Calculated forecasted rate > RATE_DECR_CYCLE

     If both equations are true, the rate change during the repricing period has exceeded RATE_DECR_CYCLE. In this case, the new forecasted rate is limited to the previous current rate - RATE_DECR_CYCLE.

     This is illustrated in the following example:

     RATE_DECR_CYCLE = 2.00 (200 basis points)

     Previous Current Rate = 10.00

     Calculated (raw) rate = 7.75

     - 10.00 > 7.75
     - 10.00 - 7.75 (= 2.25) > 2.00

     New current rate = 10.00 - 2.00 = 8.00

Data Verification Requirements and Suggested Defaults

- For fixed-rate accounts, RATE_DECR_CYCLE = 0.
- For variable-rate accounts without a maximum rate decrease, RATE_DECR_CYCLE = 0.
- For RATE_DECR_CYCLE <> 0, 0 <= RATE_DECR_CYCLE <= RATE_DECR_LIFE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Floor Life (RATE_FLOOR_LIFE)

Definition

Minimum interest rate allowed during the life of an adjustable-rate instrument.

Module Usage

Oracle Transfer Pricing references RATE_FLOOR_LIFE when calculating option costs.

The cash flow engine references RATE_FLOOR_LIFE when calculating a forecasted interest rate for adjustable-rate records.

1. For adjustable-rate records, the cash flow engine matches the REPRICE_ FREQ, INTEREST_RATE_CODE, and the reprice date to the information generated by the Monte Carlo rate engine. The margin is then added to this forecasted rate. Any rounding, rate caps/floors (RATE_FLOOR_ LIFE), and tease periods are applied, and the resulting rate is applied to the record as the repriced rate.
2. If the rate plus margin < RATE_FLOOR_LIFE, the cash flow engine sets the forecasted rate of the record equal to the RATE_FLOOR_LIFE. Rounding and tease periods are then applied. For details of the repricing process see, Next Repricing Date (NEXT_REPRICE_DATE), page 7-56.
3. For any forecasted rate changes throughout the life of the instrument, the cash flow engine references RATE_FLOOR_LIFE.

Data Verification Requirements and Suggested Defaults

- If the record is fixed-rate, RATE_FLOOR_LIFE = 0.
- If the record is adjustable without a rate floor, RATE_FLOOR_LIFE = 0.

- If RATE_FLOOR_LIFE <> 0, CUR_GROSS_RATE and CUR_NET_RATE >= RATE_FLOOR_LIFE.
- If RATE_FLOOR_LIFE <> 0, RATE_FLOOR_LIFE = ORG_RATE - RATE_DECR_LIFE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Increase Limit - Cycle (RATE_INCR_CYCLE)

Definition

Maximum rate increase allowed during a repricing cycle for an adjustable-rate instrument.

Module Usage

Oracle Transfer Pricing references RATE_INCR_CYCLE when calculating option costs.

The cash flow engine references RATE_INCR_CYCLE when calculating a forecasted interest rate. RATE_INCR_CYCLE sets the maximum amount (in terms of basis points) that the interest rate may increase in a given REPRICE_FREQ.

1. For adjustable-rate records, the cash flow engine matches the REPRICE_ FREQ, INTEREST_RATE_CODE, and the reprice date to the information generated by the Monte Carlo rate engine. The margin will then be added to this forecasted rate. Any rounding, rate caps (RATE_INCR_CYCLE) or floors, and tease periods are applied, and the resulting rate is applied to the record as its repriced rate.

   Note: ADJUSTABLE_TYPE_CODE = 30 or 50 does not reference reprice dates.

2. When applying the RATE_INCR_CYCLE, the engine checks for the following:
   - Calculated forecasted rate > Previous Current Rate
   - Calculated forecasted rate - Previous Current Rate > RATE_INCR_CYCLE.

     If both equations are true, the rate change during the repricing period has exceeded RATE_INCR_CYCLE. In this case, the new forecasted rate is limited to the previous current rate + RATE_INCR_CYCLE. This is illustrated in the following example:

     RATE_INCR_CYCLE = 2.00 (200 basis points)

Previous Current Rate = 10.00

Calculated Rate = 12.25

- 12.25 > 10.00
- 12.25 -10.00 (= 2.25) > 2.00

New Current Rate = 10.00 + 2.00 = 12.00

Data Verification Requirements and Suggested Defaults

- For fixed-rate accounts, RATE_INCR_CYCLE = 0.
- For variable-rate accounts without a maximum rate increase, RATE_INCR_CYCLE = 0.
- For RATE_INCR_CYCLE <> 0, 0 <= RATE_INCR_CYCLE <= RATE INCR LIFE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Set Lag (RATE_SET_LAG)

Definition

Used in conjunction with RATE_SET_LAG_MULT to define the period by which repricing lags the current interest rate changes.

Module Usage

Oracle Transfer Pricing references RATE_SET_LAG when calculating option costs.

The cash flow engine references RATE_SET_LAG when calculating a forecasted interest rate.

1. For adjustable-rate records, the cash flow engine matches the REPRICE_ FREQ, INTEREST_RATE_CODE, and the lagged reprice date (after referencing RATE_SET_LAG) to the information generated by the Monte Carlo rate engine. The margin is then added to this forecasted rate. Any rounding, rate caps/floors, and tease periods are applied, and the resulting rate is applied to the record as its repriced rate.

   Note: ADJUSTABLE_TYPE_CODE = 30 or 50 does not reference

reprice dates.

2. If RATE_SET_LAG > 0, the cash flow engine does not assign a forecasted interest rate based upon NEXT_REPRICE_DATE. Instead, the engine assigns the account an interest rate based upon the date NEXT_ REPRICE_DATE - RATE_SET_LAG.

   This is illustrated in the following example:

   REPRICE_FREQ = 3

   REPRICE_FREQ_MULT = M

   NEXT_REPRICE_DATE = 11/30/2007

   RATE_SET_LAG = 1 RATE_SET_LAG_ MULT = M

   MARGIN = 1.00

   In this example, the account is tied to the Treasury Yield Curve. Due to the RATE_SET_LAG, the cash flow engine reference the Treasury Yield Curve one month before the NEXT_REPRICE_DATE. The 3-month point on the Treasury Yield Curve on 10/31/2007 equals 5%. Therefore, the repriced rate equals 6% (5% plus the 1% margin).

3. If the RATE_SET_LAG > (Cash Flow Date - CALENDAR_PERIOD), the cash flow engine uses the base rate from the Monte Carlo rate engine. The cash flow engine does not reference the rates historically. For instance, if a 3-month RATE_SET_LAG is applied to a repricing event that is two months from the CALENDAR_PERIOD, the cash flow engine does not reference the Historical Rates to obtain the yield curve information that is one month before the CALENDAR_PERIOD. Instead, the cash flow engine applies the base rate from the Monte Carlo rate engine output.

Data Verification Requirements and Suggested Defaults

- RATE_SET_LAG must be either 0 or a positive value.
- If the record is a fixed-rate account, RATE_SET_LAG = 0.
- If the record is an adjustable-rate account, RATE_SET_LAG >= 0

  Note: Validation of RATE_SET_LAG should always be done in conjunction with RATE_SET_LAG_MULT.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Rate Set Lag Multiplier (RATE_SET_LAG_MULT)

Definition

Used in conjunction with RATE_SET_LAG to define the rate set lag period.

Module Usage

Oracle Transfer Pricing references RATE_SET_LAG_MULT when calculating option costs. RATE_SET_LAG_MULT determines the units (months, days, or years) of RATE_SET_LAG.

Data Verification Requirements and Suggested Defaults

- Valid values are:
  - D: Days
  - M: Months
  - Y: Years
- For information on RATE_SET_LAG_MULT validation, Rate Set Lag (RATE_SET_LAG), page 7-84.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Remaining Number of Payments (REMAIN_NO_PMTS)

Definition

The remaining number of principal, interest, or principal and interest payments to be made from the CALENDAR_PERIOD until the MATURITY_DATE of the record.

Module Usage

The Oracle Transfer Pricing cash flow methodologies reference REMAIN_NO_PMTS when calculating and processing payments. The cash flow engine uses REMAIN_NO_PMTS to determine the number of payments that remain to be paid until the account matures.

The number of remaining payments is used by the cash flow engine for two purposes:

- **Definition of payment dates on which principal and interest is to be paid:** As

each payment is made during the life of the instrument, the cash flow engine reduces the REMAIN_NO_PMTS by 1 and rolls the payment period forward by the PMT_FREQ. If the newly calculated REMAIN_NO_PMTS = 1, the cash flow engine no longer rolls the PMT_FREQ and makes the next (and final) payment on the MATURITY_DATE. See: Amortization Term (AMRT_TERM), page 7-16.

- **Amortization recalculation purposes:** When a recalculation occurs (repricing or negative amortization event), the cash flow engine references the REMAIN_NO_PMTS for recalculation of amortization. See: Current Payment (CUR_PAYMENT), page 7-26.

  There are some exceptions to the use of REMAIN_NO_PMTS for balloon records and specific user-defined payment patterns and schedules. Depending on the record characteristics, the cash flow engine may calculate the remaining number of payments itself. See: Amortization Term (AMRT_TERM), page 7-16.

1. The Transfer Pricing Remaining Term Pricing Basis cash flow methodology for fixed-rate records uses REMAIN_NO_PMTS as described above.

2. The Transfer Pricing Standard Pricing Basis cash flow methodology for an adjustable-rate record calculates the remaining number of payments as follows:

   REMAIN_NO_PMTS + number of payment periods between the NEXT_PAYMENT_DATE and the LAST_REPRICING_DATE

Data Verification Requirements and Suggested Defaults

- REMAIN_NO_PMTS >= 1.
- If REMAIN_NO_PMTS = 0, record is invalid.
- If REMAIN_NO_PMTS = 1, NEXT_PAYMENT_DATE = MATURITY_DATE.
- If REMAIN_NO_PMTS > 1, NEXT_PAYMENT_DATE < MATURITY_DATE.
- Generally:
  - REMAIN_NO_PMTS * PMT_FREQ <= ORG_TERM * ORG_TERM_MULT.
  - REMAIN_NO_PMTS * PMT_FREQ <= ISSUE_TERM * ISSUE_TERM_MULT.
  - MATURITY_DATE <= NEXT_PAYMENT_DATE + (REMAIN_NO_PMTS * PMT_FREQ).
- For non-term accounts, REMAIN_NO_PMTS = 1.
- The maximum number of payment and repricing events that can be modeled cannot exceed 2000. REMAIN_NO_PMTS + (REMAIN_TERM (in days)) /

(REPRICE_FREQ (in days)) > 2000.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Repricing Frequency (REPRICE_FREQ)

Definition

Used in conjunction with REPRICE_FREQ_MULT to define the frequency of rate change of an account.

Module Usage

Oracle Transfer Pricing references REPRICE_FREQ when identifying adjustable-rate instruments and calculating a transfer pricing term or forecasted interest rate.

1. REPRICE_FREQ is used to identify whether records are adjustable or not. If REPRICE_FREQ > 0, the record is considered adjustable. If REPRICE_FREQ = 0, the record is fixed-rate.

2. For adjustable-rate straight term methodology, Oracle Transfer Pricing matches the REPRICE_FREQ to the same term on the Transfer Pricing yield curve (Interest Rate Code) defined in the Historical Rates Rule. See: Last Repricing Date (LAST_REPRICE_DATE), page 7-41.

When Calculate Option Costs is selected on the Transfer Pricing Process rule, the following applies:

1. In identifying an adjustable record, the cash flow engine uses ADJUSTABLE_ TYPE_CODE and REPRICE_FREQ. If the ADJUSTABLE_TYPE_CODE > 0 and the REPRICE_FREQ > 0, the record is adjustable.

2. The cash flow engine uses the REPRICE_FREQ to identify repricing events beyond the NEXT_ REPRICE_DATE. The cash flow engine rolls forward from NEXT_REPRICE_DATE by the REPRICE_FREQ to define the remaining forecasted reprice dates for the record. Rolling by the REPRICE_FREQ continues until MATURITY_DATE. See: Next Repricing Date (NEXT_REPRICE_DATE), page 7-56.

3. In defining the CUR_NET_RATE and CUR_GROSS_RATE on each reprice date, the cash flow engine matches the INTEREST_RATE_CODE, the reprice date, and the REPRICE_FREQ of the record to the appropriate term point on the forecasted Interest Rate Code (IRC) generated by Monte Carlo rate engine. To this derived rate, the cash flow engine adds the relevant margin amount (MARGIN, MARGIN_GROSS). Any rate caps/floors, tease periods, and rounding is then

applied.

Note: ADJUSTABLE_TYPE_CODE = 30 and 50 does not reference reprice date.

Data Verification Requirements and Suggested Defaults

- If REPRICE_FREQ > 0, MATURITY_DATE >= NEXT_REPRICE_DATE > CALENDAR PERIOD.
- If REPRICE_FREQ > 0, INTEREST_RATE_CODE is in the range of 001 - 999.
- If REPRICE_FREQ > 0, ADJUSTABLE_TYPE_CODE > 0.
- REPRICE_FREQ must be either 0 or positive.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Repricing Frequency Multiplier (REPRICE_FREQ_MULT)

Definition

Used in conjunction with REPRICE_FREQ to define the frequency of rate change of an account.

Module Usage

Oracle Transfer Pricing references REPRICE_FREQ when identifying adjustable-rate instruments and calculating a transfer pricing term or forecasted interest rate. REPRICE_FREQ_MULT determines the units (months, days, or years) of REPRICE_FREQ.

Data Verification Requirements and Suggested Defaults

Valid values are:

- D: Days
- M: Months
- Y: Years

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Teaser-rate End Date (TEASER_END_DATE)

Definition

Date that the tease rate (introductory rate) ends and the normal product interest rate begins.

Module Usage

The cash flow engine references TEASER_END_DATE when calculating a forecasted interest rate. The TEASER_END_DATE defines the end of a tease period, which is an initial low interest rate period from the origination of a loan. At the TEASER_END_DATE, the low rate is repriced to a value defined by the market index plus margin.

1. Because TEASER_END_DATE overrides the other repricing fields, if a record is currently in its tease period, the last repricing date equals the ORIGINATION_DATE and the next repricing date equals the TEASER_END_DATE.

2. If the TEASER_END_DATE is greater than the CALENDAR_PERIOD, the Oracle Transfer Pricing mid-period repricing does not apply. See: Mid-period Repricing, *Oracle Transfer Pricing User Guide*.

If calculate Option Costs is selected on the Transfer Pricing Process rule, the following applies:

1. The cash flow engine does not adjust the rate on an adjustable-rate account until the TEASER_END_DATE is less than the current date within the cash flow timeline.

2. TEASER_END_DATE takes precedence over NEXT_REPRICE_DATE and REPRICE_FREQ. Even if NEXT_REPRICE_DATE < TEASER_END_DATE, the record does not reprice until the TEASER_END_DATE.

3. TEASER_END_DATE does not apply to fixed-rate accounts.

Data Verification Requirements and Suggested Defaults

- TEASER_END_DATE > ORIGINATION_DATE .

- TEASER_END_DATE < MATURITY_DATE.

- For fixed-rate accounts TEASER_END_DATE = 01-JAN-1900.
- TEASER_END_DATE >= NEXT_REPRICE_DATE.

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Transfer Rate (TRANSFER_RATE)

Definition

The associated transfer rate for the account, calculated using the Standard transfer pricing calculation mode.

Module Usage

After calculating the transfer rate for a record using one of the transfer pricing methodologies and the Standard transfer pricing calculation mode, the result is written out to the TRANSFER_RATE column.

> **Note:** When the Remaining Term transfer pricing calculation mode is used, the transfer rate is written to the TRAN_RATE_REM_TERM field.

Data Verification Requirements and Suggested Defaults

If TRANSFER RATES are to be loaded directly from client systems, data must be in percent format (10% = 10 not 0.10).

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Transfer Rate Remaining Term (TRAN_RATE_REM_TERM)

Definition

The associated transfer rate for the account, using the Remaining Term transfer pricing calculation mode.

Module Usage

After calculating the transfer rate for a record using one of the transfer pricing

methodologies and the Remaining Term transfer pricing calculation mode, the result is written out to the TRAN_RATE_REM_TERM column.

> **Note:** When the Standard transfer pricing calculation mode is used, the transfer rate is written to the TRANSFER_RATE column.

Data Verification Requirements and Suggested Defaults

If TRAN_RATE_REM_TERM values are to be loaded directly from client systems, data must be in % format (10% = 10 not 0.10).

Related Topics

Overview of Cash Flow Columns, page 7-1

Cash Flow Column Descriptions, page 7-5

Index

A

C

D

E

F

I

L

M

N

O

P

R

S

T

V

Y