Oracle® Financial Services Accounting Hub

Implementation Guide

Release 12.1

Part No. E13420-06

October 2016

ORACLE

Oracle Financial Services Accounting Hub Implementation Guide, Release 12.1

Part No. E13420-06

Primary Author: Sudha Seshadri

Contributing Author: Ayse Aba, Robert Anderson, Mizuru Asada, Kaouther Boussema-Ghalem, Wynne Chan, Ananya Chandra, Miranda Ho, Shishir Joshi, Jorge Larre-Borges, Elaine Lau, Julianna Litwin, Raj Maisuria, Rian Monnahan, Noela Nakos, Luc Nayrolles, Ivan Pena, Neil Ramsay, Johanna Rusly, Dimple Shah, Esha Sen, Wei Shen, Sandeep Singhania, Matthew Skurdahl, Alison Wan

Contents

2 Analysis of Accounting Events

3 Transaction Objects Analysis

4 Seeding Event Information Using Accounting Methods Builder

5 Create and Assign Sources

6 Revise Source Definitions and Assign Accounting Attributes

7 Financial Services Accounting Hub Seed Data

8 Seeding Calls to Subledger Event Capture Routines

9 Accounting Program and Period Close Integration

10 Gapless Event Creation and Processing

11 Financial Services Accounting Hub Security

12 Manual Subledger Journal Entries API

13 Drilldown

14 Financial Services Accounting Hub Repository Specification

15 Calculation of Accounted and Gain or Loss Amounts

16 Transaction Account Builder

17 Application Accounting Definitions Loader

Index

Send Us Your Comments

Oracle Financial Services Accounting Hub Implementation Guide, Release 12.1

Part No. E13420-02

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 My Oracle Support 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 12.1 of the *Oracle Financial Services Accounting Hub Implementation 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 xiv for more Oracle Applications product information.

TTY Relay Access to Oracle Support Services

To reach AT&T Customer Assistants, dial 711 or 1.800.855.2880. An AT&T Customer Assistant will relay information between the customer and Oracle Support Services at 1.800.223.1711. Complete instructions for using the AT&T relay services are available at http://www.consumer.att.com/relay/tty/standard2.html. After the AT&T Customer Assistant contacts Oracle Support Services, an Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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

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 Oracle*MetaLink.*

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 book is intended for use by anyone who is responsible for installing or upgrading Oracle Applications. It provides instructions for running Rapid Install either to carry out a fresh installation of Oracle Applications Release 12, or as part of an upgrade from Release 11i to Release 12. The book also describes the steps needed to install the technology stack components only, for the special situations where this is applicable.

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 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 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 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 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 Advanced Global Intercompany System User's Guide:

This guide describes the self service application pages available for Intercompany users. It includes information on setting up intercompany, entering intercompany transactions, importing transactions from external sources and generating reports.

Oracle Advanced Collections User Guide:

This guide describes how to use the features of Oracle Advanced Collections to manage your collections activities. It describes how collections agents and managers can use Oracle Advanced Collections to identify delinquent customers, review payment history and aging data, process payments, use strategies and dunning plans to automate the collections process, manage work assignments, and handle later-stage delinquencies.

Oracle Advanced Collections Implementation Guide:

This guide describes how to configure Oracle Advanced Collections and its integrated products. It contains the steps required to set up and verify your implementation of Oracle Advanced Collections.

Oracle Applications Multiple Organizations Implementation Guide:

This guide describes the multiple organizations concepts in Oracle Applications. It describes in detail on setting up and working effectively with multiple organizations in Oracle Applications.

Oracle Assets User Guide:

This guide provides you with information on how to implement and use Oracle Assets. Use this guide to understand the implementation steps required for application use, including defining depreciation books, depreciation method, and asset categories. It also contains information on setting up assets in the system, maintaining assets, retiring and reinstating assets, depreciation, group depreciation, accounting and tax accounting, budgeting, online inquiries, impairment processing, and Oracle Assets reporting. The guide explains using Oracle Assets with Multiple Reporting Currencies (MRC). This guide also includes a comprehensive list of profile options that you can set to customize application behavior.

Oracle Balanced Scorecard User Guide:

This guide describes how to use Oracle Balanced Scorecard to manage performance. It contains information on how to use scorecard views and objective reports.

Oracle Balanced Scorecard Administrator Guide:

This guide describes how to set up and administer Oracle Balanced Scorecard and scorecard systems. For scorecard designers, this guide explains how to design and prototype scorecards and measures. It also explains how to move scorecards into production. For administrators, this guide explains how to generate the database schema; load data; manage user and scorecard security; and migrate scorecards to other instances.

Oracle Balanced Scorecard Install Guide:

This guide describes how to how to install the Balanced Scorecard Architect components.

Oracle Bill Presentment Architecture User Guide:

This guide provides you information on using Oracle Bill Presentment Architecture. Consult this guide to create and customize billing templates, assign a template to a rule and submit print requests. This guide also provides detailed information on page references, seeded content items and template assignment attributes.

Oracle Cash Management User Guide:

This guide describes how to use Oracle Cash Management to clear your receipts, as well as reconcile bank statements with your outstanding balances and transactions. This manual also explains how to effectively manage and control your cash cycle. It provides

comprehensive bank reconciliation and flexible cash forecasting.

Oracle Credit Management User Guide:

This guide provides you with information on how to use Oracle Credit Management. This guide includes implementation steps, such as how to set up credit policies, as well as details on how to use the credit review process to derive credit recommendations that comply with your credit policies. This guide also includes detailed information about the public application programming interfaces (APIs) that you can use to extend Oracle Credit Management functionality.

Oracle Customer Data Librarian User Guide:

This guide describes how to use Oracle Customer Data Librarian to establish and maintain the quality of the Trading Community Architecture Registry, focusing on consolidation, cleanliness, and completeness. Oracle Customer Data Librarian has all of the features in Oracle Customers Online, and is also part of the Oracle Customer Data Management product family.

Oracle Customer Data Librarian Implementation Guide:

This guide describes how to implement Oracle Customer Data Librarian. As part of implementing Oracle Customer Data Librarian, you must also complete all the implementation steps for Oracle Customers Online.

Oracle Customers Online User Guide:

This guide describes how to use Oracle Customers Online to view, create, and maintain your customer information. Oracle Customers Online is based on Oracle Trading Community Architecture data model and functionality, and is also part of the Oracle Customer Data Management product family.

Oracle Customers Online Implementation Guide:

This guide describes how to implement Oracle Customers Online.

Oracle Daily Business Intelligence Implementation Guide:

This guide describes how to implement Oracle Daily Business Intelligence, including information on how to create custom dashboards, reports, and key performance indicators.

Oracle Daily Business Intelligence User Guide:

This guide describes how to use the preseeded Daily Business Intelligence dashboards, reports, and key performance indicators.

Oracle E-Business Suite Diagnostics User's Guide

This manual contains information on implementing, administering, and developing diagnostics tests in the Oracle E-Business Suite Diagnostics framework.

Oracle E-Business Suite Integrated SOA Gateway User's Guide

This guide describes the high level service enablement process, explaining how users can browse and view the integration interface definitions and services residing in

Oracle Integration Repository.

Oracle E-Business Suite Integrated SOA Gateway Implementation Guide

This guide explains how integration repository administrators can manage and administer the service enablement process (based on the service-oriented architecture) for both native packaged public integration interfaces and composite services (BPEL type). It also describes how to invoke Web services from Oracle E-Business Suite by employing the Oracle Workflow Business Event System; how to manage Web service security; and how to monitor SOAP messages.

Oracle E-Business Suite Integrated SOA Gateway Developer's Guide

This guide describes how system integration developers can perform end-to-end service integration activities. These include orchestrating discrete Web services into meaningful end-to-end business processes using business process execution language (BPEL), and deploying BPEL processes at run time.

It also explains in detail how to invoke Web services using the Service Invocation Framework. This includes defining Web service invocation metadata, invoking Web services, managing errors, and testing the Web service invocation.

Oracle E-Business Tax User Guide:

This guide describes the entire process of setting up and maintaining tax configuration data, as well as applying tax data to the transaction line. It describes the entire regime-to-rate setup flow of tax regimes, taxes, statuses, rates, recovery rates, tax jurisdictions, and tax rules. It also describes setting up and maintaining tax reporting codes, fiscal classifications, tax profiles, tax registrations, configuration options, and third party service provider subscriptions. You also use this manual to maintain migrated tax data for use with E-Business Tax.

Oracle E-Business Tax Implementation Guide:

This guide provides a conceptual overview of the E-Business Tax tax engine, and describes the prerequisite implementation steps to complete in other applications in order to set up and use E-Business Tax. The guide also includes extensive examples of setting up country-specific tax requirements.

Oracle E-Business Tax Reporting Guide:

This guide explains how to run all tax reports that make use of the E-Business Tax data extract. This includes the Tax Reporting Ledger and other core tax reports, country-specific VAT reports, and Latin Tax Engine reports.

Oracle E-Business Tax: Vertex Q-Series and Taxware Sales/Use Tax System Implementation Guide

This guide explains how to setup and use the services of third party tax service providers for US Sales and Use tax. The tax service providers are Vertex Q-Series and Taxware Sales/Use Tax System. When implemented, the Oracle E-Business Tax service subscription calls one of these tax service providers to return a tax rate or amount whenever US Sales and Use tax is calculated by the Oracle E-Business Tax tax engine.

This guide provides setup steps, information about day-to-day business processes, and a technical reference section.

Oracle Embedded Data Warehouse User Guide:

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

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 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 Enterprise Planning and Budgeting User's Guide:

This guide describes Enterprise Planning and Budgeting, which is an enterprise application that provides rich functionality to control the business processes of planning, budgeting, and forecasting. Enterprise Planning and Budgeting is deployed as a Web based solution using the power of Oracle relational technology to deliver scalable, multi-dimensional analysis and monitoring.

Oracle Financial Consolidation Hub User Guide:

This guide describes how to set up, maintain, and troubleshoot Oracle Financial Consolidation Hub. It describes setting up entities, categories, consolidation methods, consolidation rules, intercompany rules, calendar maps, translation, consolidation hierarchies, analytical reporting, and the Excel add-in. The guide also includes chapters on submitting data, running consolidations, accounting for acquisitions and disposals, integrating with Internal Controls Manager and WebADI spreadsheets.

Oracle Financial Services Reference Guide:

This guide provides reference material for Oracle Financial Services applications in Release 12, such as Oracle Transfer Pricing, and includes technical details about application use as well as general concepts, equations, and calculations.

Oracle Financial Services Implementation Guide:

This guide describes how to set up Oracle Financial Services 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 Financials and Oracle Procurement Functional Upgrade Guide: Release 11i to Release 12:

This guides provides detailed information about the functional impacts of upgrading Oracle Financials and Oracle Procurement products from Release 11i to Release 12. This guide supplements the *Oracle Applications Upgrade Guide: Release 11i to Release 12*.

Oracle Financials Concepts Guide:

This guide describes the fundamental concepts of Oracle Financials. The guide is intended to introduce readers to the concepts used in the applications, and help them compare their real world business, organization, and processes to those used in the applications.

Oracle Financials Country-Specific Installation Supplement:

This guide provides general country information, such as responsibilities and report security groups, as well as any post-install steps required by some countries.

Oracle Financials for the Americas User Guide:

This guide describes functionality developed to meet specific business practices in countries belonging to the Americas region. Consult this user guide along with your financial product user guides to effectively use Oracle Financials in your country.

Oracle Financials for Asia/Pacific User Guide:

This guide describes functionality developed to meet specific business practices in countries belonging to the Asia/Pacific region. Consult this user guide along with your financial product user guides to effectively use Oracle Financials in your country.

Oracle Financials for Europe User Guide:

This guide describes functionality developed to meet specific business practices in countries belonging to the European region. Consult this user guide along with your financial product user guides to effectively use Oracle Financials in your country.

Oracle Financials for India User Guide:

This guide provides information on how to use Oracle Financials for India. Use this guide to learn how to create and maintain setup related to India taxes, defaulting and calculation of taxes on transactions. This guide also includes information about accounting and reporting of taxes related to India.

Oracle Financials for India Implementation Guide:

This guide provides information on how to implement Oracle Financials for India. Use this guide to understand the implementation steps required for application use, including how to set up taxes, tax defaulting hierarchies, set up different tax regimes, organization and transactions.

Oracle Financials Glossary:

The glossary includes definitions of common terms that are shared by all Oracle

Financials products. In some cases, there may be different definitions of the same term for different Financials products. If you are unsure of the meaning of a term you see in an Oracle Financials guide, please refer to the glossary for clarification. You can find the glossary in the online help or in the *Oracle Financials Implementation Guide*.

Oracle Financials Implementation Guide:

This guide provides information on how to implement the Oracle Financials E-Business Suite. It guides you through setting up your organizations, including legal entities, and their accounting, using the Accounting Setup Manager. It covers intercompany accounting and sequencing of accounting entries, and it provides examples.

Oracle Financials RXi Reports Administration Tool User Guide:

This guide describes how to use the RXi reports administration tool to design the content and layout of RXi reports. RXi reports let you order, edit, and present report information to better meet your company's reporting needs.

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 Incentive Compensation Implementation Guide:

This guide provides Compensation Administrators with guidance during implementation of Oracle Incentive Compensation. The procedures are presented in the recommended order that they should be performed for successful implementation. Appendixes are included that describe system profiles, lookups, and other useful information.

Oracle Incentive Compensation User Guide:

This guide helps Compensation Managers, Compensation Analysts, and Plan administrators to manage Oracle Incentive Compensation on a day-to-day basis. Learn how to create and manage rules hierarchies, create compensation plans, collect transactions, calculate and pay commission, and use Sales Credit Allocation.

Oracle Internal Controls Manager Implementation Guide:

This guide describes implementation information for Oracle Internal Controls Manager, a comprehensive tool for executives, controllers, internal audit departments, and public accounting firms to document and test internal controls and monitor ongoing

compliance. It is based on COSO (Committee of Sponsoring Organizations) standards.

Oracle Internet Expenses Implementation and Administration Guide:

This book explains in detail how to configure Oracle Internet Expenses and describes its integration with other applications in the E-Business Suite, such as Oracle Payables and Oracle Projects. Use this guide to understand the implementation steps required for application use, including how to set up policy and rate schedules, credit card policies, audit automation, and the expenses spreadsheet. This guide also includes detailed information about the client extensions that you can use to extend Oracle Internet Expenses functionality.

Oracle iAssets User Guide

This guide provides information on how to implement and use Oracle iAssets. Use this guide to understand the implementation steps required for application use, including setting up Oracle iAssets rules and related product setup steps. It explains how to define approval rules to facilitate the approval process. It also includes information on using the Oracle iAssets user interface to search for assets, create self-service transfer requests and view notifications.

Oracle iProcurement Implementation and Administration Guide:

This manual describes how to set up and administer Oracle iProcurement. Oracle iProcurement enables employees to requisition items through a self–service, Web interface.

Oracle iReceivables Implementation Guide:

This guide provides information on how to implement Oracle iReceivables. Use this guide to understand the implementation steps required for application use, including how to set up and configure iReceivables, and how to set up the Credit Memo Request workflow. There is also a chapter that provides an overview of major features available in iReceivables.

Oracle iSupplier Portal User Guide:

This guide contains information on how to use Oracle iSupplier Portal to enable secure transactions between buyers and suppliers using the Internet. Using Oracle iSupplier Portal, suppliers can monitor and respond to events in the procure-to-pay cycle.

Oracle iSupplier Portal Implementation Guide:

This guide contains information on how to implement Oracle iSupplier Portal and enable secure transactions between buyers and suppliers using the Internet.

Oracle Loans User Guide:

This guide describes how to set up and use Oracle Loans. It includes information on how to create, approve, fund, amortize, bill, and service extended repayment plan and direct loans.

Oracle Partner Management Implementation and Administration Guide:

This guide helps Vendor administrators to set up and maintain relationships and

programs in the Partner Management application. The main areas include setting up the partner and channel manager dashboards, partner setup, partner programs and enrollment, opportunity and referral management, deal registration, special pricing management, and partner fund management.

Oracle Partner Management Vendor User Guide:

This guide assists vendor users in using Partner Management on a daily basis. This includes interaction with the partner and channel manager dashboards, working with partners and partner programs, managing opportunities and referrals, registering deals, and working with special pricing and partner funds.

Oracle Payables User Guide:

This guide describes how to use Oracle Payables to create invoices and make payments. In addition, it describes how to enter and manage suppliers, import invoices using the Payables open interface, manage purchase order and receipt matching, apply holds to invoices, and validate invoices. It contains information on managing expense reporting, procurement cards, and credit cards. This guide also explains the accounting for Payables transactions.

Oracle Payables Implementation Guide:

This guide provides you with information on how to implement Oracle Payables. Use this guide to understand the implementation steps required for how to set up suppliers, payments, accounting, and tax.

Oracle Payables Reference Guide:

This guide provides you with detailed information about the Oracle Payables open interfaces, such as the Invoice open interface, which lets you import invoices. It also includes reference information on purchase order matching and purging purchasing information.

Oracle Payments Implementation Guide:

This guide describes how Oracle Payments, as the central payment engine for the Oracle E-Business Suite, processes transactions, such as invoice payments from Oracle Payables, bank account transfers from Oracle Cash Management, and settlements against credit cards and bank accounts from Oracle Receivables. This guide also describes how Oracle Payments is integrated with financial institutions and payment systems for receipt and payment processing, known as funds capture and funds disbursement, respectively. Additionally, the guide explains to the implementer how to plan the implementation of Oracle Payments, how to configure it, set it up, test transactions, and how use it with external payment systems.

Oracle Payments User Guide:

This guide describes how Oracle Payments, as the central payment engine for the Oracle E-Business Suite, processes transactions, such as invoice payments from Oracle Payables, bank account transfers from Oracle Cash Management, and settlements against credit cards and bank accounts from Oracle Receivables. This guide also describes to the Payment Administrator how to monitor the funds capture and funds

disbursement processes, as well as how to remedy any errors that may arise.

Oracle Procurement Buyer's Guide to Punchout and Transparent Punchout:

This guide contains necessary information for customers implementing remote catalog content on a supplier's Web site or on Oracle Exchange.

Oracle Procurement Contracts Online Help:

This guide is provided as online help only from the Oracle Procurement Contracts application and includes information about creating and managing your contract terms library.

Oracle Procurement Contracts Implementation and Administration Guide:

This guide describes how to set up and administer Oracle Procurement Contracts. Oracle Procurement Contracts enables employees to author and maintain complex contracts through a self–service, Web interface.

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 Public Sector Financials User Guide:

This guide describes how to set up and administer Oracle Public Sector Advanced Features. It describes Encumbrance Reconciliation Reports, GASB 34/35 Asset Accounting, and Funds Available Enhancements.

Oracle Purchasing User's Guide:

This guide describes how to create and approve purchasing documents, including requisitions, different types of purchase orders, quotations, RFQs, and receipts. This guide also describes how to manage your supply base through agreements, sourcing rules, and approved supplier lists. In addition, this guide explains how you can automatically create purchasing documents based on business rules through integration with Oracle Workflow technology, which automates many of the key procurement processes.

Oracle Receivables User Guide:

This guide provides you with information on how to use Oracle Receivables. Use this guide to learn how to create and maintain transactions and bills receivable, enter and apply receipts, enter customer information, and manage revenue. This guide also includes information about accounting in Receivables. Use the Standard Navigation Paths appendix to find out how to access each Receivables window.

Oracle Receivables Implementation Guide:

This guide provides you with information on how to implement Oracle Receivables. Use this guide to understand the implementation steps required for application use, including how to set up customers, transactions, receipts, accounting, tax, and collections. This guide also includes a comprehensive list of profile options that you can set to customize application behavior.

Oracle Receivables Reference Guide:

This guide provides you with detailed information about all public application programming interfaces (APIs) that you can use to extend Oracle Receivables functionality. This guide also describes the Oracle Receivables open interfaces, such as AutoLockbox which lets you create and apply receipts and AutoInvoice which you can use to import and validate transactions from other systems. Archiving and purging Receivables data is also discussed in this guide.

Oracle Sourcing Implementation and Administration Guide:

This guide contains information on how to implement Oracle Sourcing to enable participants from multiple organizations to exchange information, conduct bid and auction processes, and create and implement buying agreements. This allows professional buyers, business experts, and suppliers to participate in a more agile and accurate sourcing process.

Oracle Subledger Accounting Implementation Guide:

This guide provides setup information for Oracle Subledger Accounting features, including the Accounting Methods Builder. You can use the Accounting Methods Builder to create and modify the setup for subledger journal lines and application accounting definitions for Oracle subledger applications. This guide also discusses the reports available in Oracle Subledger Accounting and describes how to inquire on subledger journal entries.

Oracle Supplier Scheduling User's Guide:

This guide describes how you can use Oracle Supplier Scheduling to calculate and maintain planning and shipping schedules and communicate them to your suppliers.

Oracle iProcurement Implementation and Administration Guide:

This manual describes how to set up and administer Oracle iProcurement. Oracle iProcurement enables employees to requisition items through a self–service, Web interface.

Oracle Procurement Contracts Implementation and Administration Guide:

This manual describes how to set up and administer Oracle Procurement Contracts. Oracle Procurement Contracts enables employees to author and maintain complex contracts through a self–service, Web interface.

Oracle Trading Community Architecture User Guide:

This guide describes the Oracle Trading Community Architecture (TCA) and how to use features from the Trading Community Manager responsibility to create, update, enrich, and cleanse the data in the TCA Registry. It also describes how to use Resource

Manager to define and manage resources.

Oracle Trading Community Architecture Administration Guide:

This guide describes how to administer and implement Oracle Trading Community Architecture (TCA). You set up, control, and manage functionality that affects data in the TCA Registry. It also describes how to set up and use Resource Manager to manage resources.

Oracle Trading Community Architecture Reference Guide:

This guide contains seeded relationship types, seeded Data Quality Management data, D&B data elements, Bulk Import interface table fields and validations, and a comprehensive glossary. This guide supplements the documentation for Oracle Trading Community Architecture and all products in the Oracle Customer Data Management family.

Oracle Trading Community Architecture Technical Implementation Guide:

This guide explains how to use the public Oracle Trading Community Architecture application programming interfaces (APIs) and develop callouts based on Oracle Workflow Business Events System (BES). For each API, this guide provides a description of the API, the PL/SQL procedure, and the Java method, as well as a table of the parameter descriptions and validations. For each BES callout, this guide provides the name of the logical entity, its description, and the ID parameter name. Also included are setup instructions and sample code.

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.

Oracle U.S. Federal Financials User's Guide:

This guide describes the common concepts for an integrated financial management solution for federal agencies to comply with the requirements of the U.S. Federal government. It describes the product architecture and provides information on Budget Execution, Prompt Payment, Treasury payments, Third party payments, Interagency transactions, Receivables management, Federal reports, CCR Integration, and Year End Closing.

Oracle U.S. Federal Financials Implementation Guide:

This guide describes the common concepts for an integrated financial management solution for federal agencies. It includes a consolidated setup checklist by page and provides detailed information on how to set up, maintain, and troubleshoot the Federal Financial application for the following functional areas: Sub Ledger Accounting, Budget Execution, Prompt Payment, Treasury payments, Third party payments, Interagency transactions, Receivables management, Federal reports, CCR Integration, and Year End Closing.

Oracle Workflow Client Installation Guide

This guide describes how to install the Oracle Workflow Builder and Oracle XML Gateway Message Designer client components for Oracle E-Business Suite.

Oracle Projects Documentation Set

Oracle Projects Implementation Guide:

Use this manual as a guide for implementing Oracle Projects. This manual also includes appendixes covering security functions, menus and responsibilities, and profile options.

Oracle Projects Fundamentals:

Oracle Project Fundamentals provides the common foundation shared across the Oracle Projects products (Project Costing, Project Billing, Project Resource Management, Project Management, and Project Portfolio Analysis). Use this guide to learn fundamental information about the Oracle Projects solution. This guide includes a Navigation Paths appendix. Use this appendix to find out how to access each window in the Oracle Projects solution.

Oracle Project Costing User Guide:

Use this guide to learn detailed information about Oracle Project Costing. Oracle Project Costing provides the tools for processing project expenditures, including calculating their cost to each project and determining the GL accounts to which the costs are posted.

Oracle Project Billing User Guide:

This guide shows you how to use Oracle Project Billing to define revenue and invoicing rules for your projects, generate revenue, create invoices, and integrate with other Oracle Applications to process revenue and invoices, process client invoicing, and measure the profitability of your contract projects.

Oracle Project Management User Guide:

This guide shows you how to use Oracle Project Management to manage projects through their lifecycles -- from planning, through execution, to completion.

Oracle Project Portfolio Analysis User Guide:

This guide contains the information you need to understand and use Oracle Project Portfolio Analysis. It includes information about project portfolios, planning cycles, and metrics for ranking and selecting projects for a project portfolio.

Oracle Project Resource Management User Guide:

This guide provides you with information on how to use Oracle Project Resource Management. It includes information about staffing, scheduling, and reporting on project resources.

Oracle Projects Glossary:

This glossary provides definitions of terms that are shared by all Oracle Projects applications. If you are unsure of the meaning of a term you see in an Oracle Projects guide, please refer to the glossary for clarification. You can find the glossary in the online help for Oracle Projects, and in the Oracle Projects Fundamentals book.

Oracle Grants Accounting Documentation

Oracle Grants Accounting User Guide:

This guide provides you with information about how to implement and use Oracle Grants Accounting. Use this guide to understand the implementation steps required for application use, including defining award types, award templates, allowed cost schedules, and burden set up. This guide also explains how to use Oracle Grants Accounting to track grants and funded projects from inception to final reporting.

Oracle Property Manager Documentation

Oracle Property Manager User Guide:

Use this guide to learn how to use Oracle Property Manager to create and administer properties, space assignments, and lease agreements.

Oracle Property Manager Implementation Guide:

Use this guide to learn how to implement Oracle Property Manager and perform basic setup steps such as setting system options and creating lookup codes, contacts, milestones, grouping rules, term templates, and a location hierarchy. This guide also describes the setup steps that you must complete in other Oracle applications before you can use Oracle Property Manager.

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

Overview of the Financial Services Accounting Hub for Implementers

Financial Services Accounting Hub Solutions

The Financial Services Accounting Hub includes the following elements:

Accounting Representations Closely Tied to the Originating Transactions

By providing an accounting representation closely tied to the originating transaction, the subledger accounting solution facilitates drill-down, third party control accounts, subledger reporting, and supporting references. Using the Financial Services Accounting Hub, companies can comply with fiscal subledger accounting and detailed audit regulations in countries where accounting must be supported by the source document.

> **Note:** Supporting references can be used to store additional source information about a subledger journal entry either at the header or line level. They can also be used to establish a subledger balance for a particular source, for a particular account.

See:Accounting Representations, page 1-2

Flexible Application Accounting Definitions

Application accounting definitions are the journal entry setups including the following that are used to create subledger journal entries:

- journal entry descriptions
- journal line types
- account derivation rules
- supporting references

Flexible application accounting definitions enable you to control the manner in which subledger transactions are accounted. By providing this flexibility, the Financial Services Accounting Hub enables enterprises to tailor subledger journal entries to both local and vertical market needs. Standard cross-application features enable you to define and distribute application accounting definitions for different accounting standards and regulations.

See: Application Accounting Definitions, *Oracle Subledger Accounting Implementation Guide*

Multiple Representations

Multiple representations enable companies to maintain corporate global subledger accounting standards, while at the same time complying with detailed local regulations. For example, companies can create standardized financial reports in the presence of accounting standards and regulations that vary by region or country.

Scalability

Scalability enables users and others outside application development to create and distribute new application accounting definitions. Enterprises can rapidly define and distribute new or modified subledger application accounting definitions to meet specific and changing requirements for their accounting.

Accounting Representations

Typically, an accounting representation refers to all journal entries that are stored in a ledger. A ledger's subledger accounting method, accounting currency, calendar, and chart of accounts determine the nature of the accounting representation.

For example, the French Fiscal accounting representation refers to all journal entries stored in one of the ledgers created for the firm's French subsidiary. The French Fiscal accounting representation is generated using the French Fiscal subledger accounting method, Euro accounting currency, the French fiscal calendar, and French Fiscal chart of

accounts.

The same firm can have a corporate accounting representation that refers to all journal entries stored in a separate corporate ledger. The corporate accounting representation is generated using the standard accrual subledger accounting method, U.S. dollar accounting currency, the corporate fiscal calendar, and a corporate chart of accounts.

Using the Accounting Methods Builder (AMB), you can create accounting representations for multiple ledgers.

> **Note:** Using the AMB, you can create and modify subledger application accounting definitions. Group these definitions into subledger accounting methods and assign them collectively to a ledger.

See: Subledger Accounting Methods, *Oracle Subledger Accounting Implementation Guide*

Introduction to Events and Sources

Event Model

Accounting events represent transactions that have a financial accounting impact. Examples of accounting events are issuing an invoice and disposing an asset. Financial accounting information can be recorded for these events. Accounting events cannot be compared to system events and programs that update transaction tables; instead they should be analyzed from a business perspective. Events are captured when transactions are committed in the subledgers.

As an example, a Payables invoice is created, then approved, possibly adjusted, and then paid or canceled. The accounting events representing these transactions can create one or more subledger journal entries and subsequently link the originating transaction to its corresponding journal entries.

Accounting events are categorized into event types. Event types are grouped into event classes that in turn are grouped into event entities. These groupings play a prominent role in the setup of the AMB. The definition of several components in the AMB is by event class or event type.

See:

- Accounting Methods Builder (AMB) Introduction, *Oracle Subledger Accounting Implementation Guide*

Sources and Source Data

Sources are a key component of the AMB. They are defined as the appropriate contextual and reference data of transactions. All contextual and reference data of transactions that are set up as sources can be used in an application accounting

definition.

The sources that are most distinctly available are those that are purely accounting in nature. These include items such as Accounting Flexfields entered on transactions, currency codes, and currency amounts. Sources that are less obviously required for application accounting definitions include items that are related to the transaction, but not necessarily only for accounting purposes. These can include items such as the asset location, vendor information, or an organization used to book a Project Accounting expenditure batch.

With the AMB, users create custom application accounting definitions that determine how the transaction is accounted. Accounting definitions in turn use source values to create the accounting for a subledger transaction.

Implementers therefore need to furnish a broad variety of sources for maximum flexibility in creating definitions for subledger journal entries. Seeded sources are provided as part of the startup data of the application.

The Financial Services Accounting Hub differentiates between sources and source values. Sources are defined (seeded) in the AMB. This is a predefined step which must be undertaken before the Financial Services Accounting Hub can be used to create journal entries. Source values are stored in the transaction objects and the Financial Services Accounting Hub uses them to create journal entries for the events.

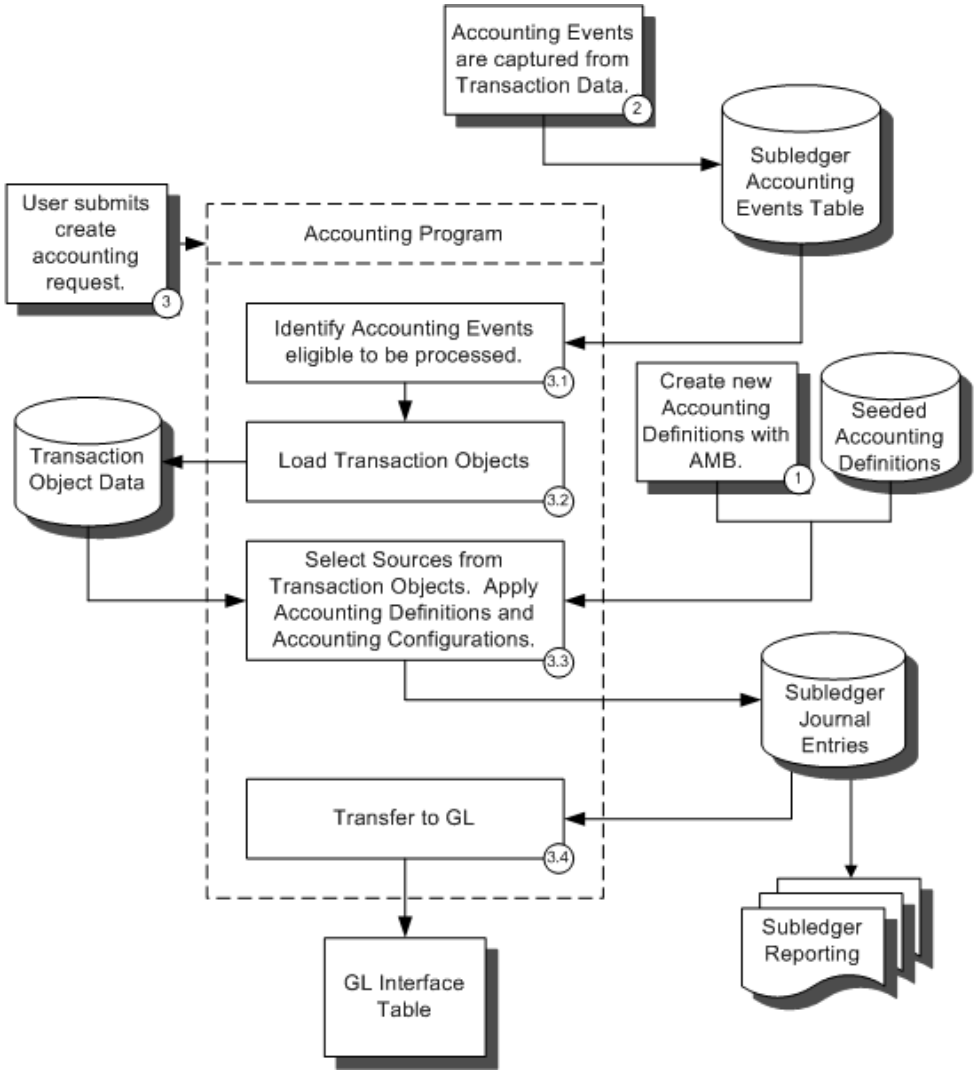
To enable the creation of subledger journal entries, implementers must complete seeding the sources in AMB and then storing source values in the transaction objects, which can be tables or views.

See: Introduction to Transaction Objects, page 3-1

Overview of the Financial Services Accounting Hub

The Overview of the Financial Services Accounting Hub figure below provides a high-level overview of the Financial Services Accounting Hub process used to create subledger journal entries and is described in the succeeding text.

Financial Services Accounting Hub Process Overview



Note: The enclosed numbers provide a chronological order to the diagram.

The above diagram illustrates the process used to create subledger journal entries.

1. The Financial Services Accounting Hub uses the AMB tool to create and modify subledger application accounting definitions.

   In conjunction with these application accounting definitions, the Accounting Program uses the transaction objects data to create subledger journal entries. For example, if an application accounting definition specifies that the customer name should appear in the description of a subledger journal entry line, then the customer name is taken from the data provided by the transaction objects.

2. When transactions are committed in a subledger, accounting events are captured and stored in the Financial Services Accounting Hub.

   The Accounting Program identifies all accounting events eligible to be processed. For each of these events, the transaction objects process provides the Accounting Program with transaction objects data (source information). This is the contextual data of the transaction, such as amounts and GL dates.

3. When the accounting program is run, application accounting definitions and accounting transaction objects data are applied to transactions to create subledger journal entries.

   Subsequently, these entries can be summarized and transferred to General Ledger.

Financial Services Accounting Hub Uptake Process

Implementers seed startup subledger application accounting definitions. Users can use the seeded definitions, copy and modify them, or create their own definitions. These definitions are applied to accounting event information to create subledger journal entries.

Before creating or modifying application accounting definitions using the AMB, several components of the Financial Services Accounting Hub need to be set up. They include the following:

- Accounting events
- Source information
- Default application accounting definitions

These initial setup steps are referred to as the Financial Services Accounting Hub uptake process.

The uptake process serves as the foundation on which other elements, such as new application accounting definitions of the architecture are constructed. While the Financial Services Accounting Hub provides the mechanisms for these elements, all implementers using the architecture need to complete the uptake process for their applications before users can create or modify application accounting definitions.

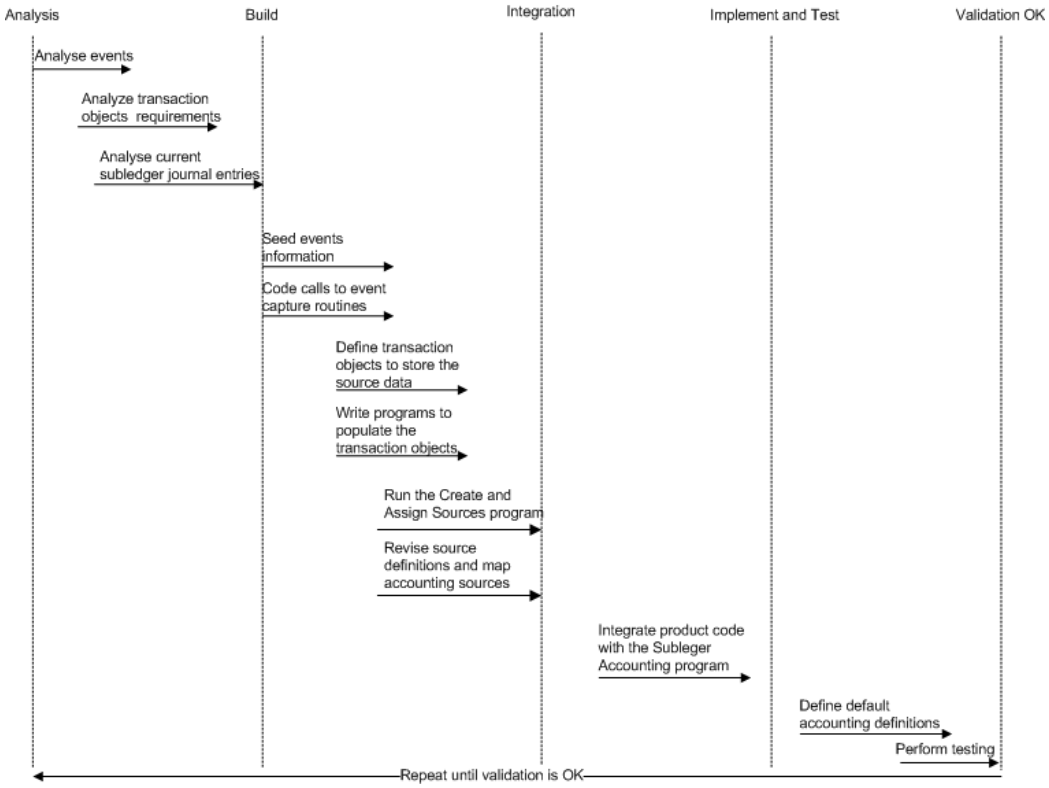
Financial Services Accounting Hub uptake involves the tasks described in the following sections and are fully described in successive chapters:

- Summary Listing of Uptake Steps, page 1-7
- Analysis, page 1-9
- Definition and Build, page 1-11
- Accounting Program Integration, page 1-15

- Implement and Test, page 1-15

Summary Listing of Uptake Steps

The diagram below shows the order of tasks to be performed when implementing the uptake of the Financial Services Accounting Hub.



The table below lists the steps shown in the diagram above and are described in the succeeding sections. Complete these steps in the order shown to implement the uptake of a subledger application to the Financial Services Accounting Hub.

Uptake Steps

Step No.	Description
	Analysis
1.	Analyze accounting events to determine what events to capture for the subledger application.

Step No.	Description
2.	Analyze transaction objects requirements. These requirements determine what transactional or environmental information (source data) should be captured as well as the transaction objects data model and objects (tables and views) to store this data.
3.	Analyze and map the application's current subledger journal entries.
	Definition and Build
4.	Register subledger applications and seed event information (event entities, event classes, process categories, and event types) in the AMB.
5.	Code calls to the Financial Services Accounting Hub event capture routines.
6.	Build transaction objects programs and objects to store the source data.
7.	Run the Create and Assign Sources program.
8.	Revise source definitions and map accounting attributes.
	Integration
9.	Modify subledger application code to integrate with the accounting program and drill-down routines.
	Implement and Test
10.	Create application accounting definitions in the AMB.
11.	Perform comprehensive testing to ensure that all accounting is correctly generated by the Financial Services Accounting Hub.

Analysis

1. Analyze Accounting Events

Some business events have financial accounting significance and require the recording of financial information. These business events are known as accounting events. The Financial Services Accounting Hub is only concerned with business events that are accounting events. Examples of business events that are accounting events, include the following:

- Issuing an invoice to a customer
- Issuing a payment to a supplier
- Retiring an asset

An accounting event and its associated transaction data typically relate to a single document or transaction.

See: Introduction to Events and Sources, page 1-3

The first task is to carry out a complete analysis to determine which events to capture. This task may need to be done in tandem with functional teams, to ensure that all possible events are accurately mapped. Event capture routines, described in Step 3. Analyze and Map Current Subledger Journal Entries, page 1-10 below, can only trap accounting events that have been identified.

Complete the following analysis to identify accounting events:

- Identify the life cycle of documents or business objects and the transactions that affect their status.
- Identify events in the life cycle that may have fiscal implications.

  All events that impact financial reports are significant and must be captured.
- Identify all business events for which contextual data and transaction amounts are available.

  This is not a mandatory requirement, but it provides maximum flexibility for creating application accounting definitions.

Oracle recommends that users thoroughly document all accounting events.

See: Introduction to Events and Sources, page 1-3

2. Analyze Transaction Objects Requirements

Sources are the appropriate contextual and reference data of transactions and are used to establish many of the items in a subledger journal entry. Complete an analysis to determine what information (source data) is necessary to successfully create subledger

journal entries from transactions.

Flexibility in creating application accounting definitions is dependent on the number of sources available. Verify that all sources that can potentially be used to create subledger journal entries are included in the accounting transaction objects. The following list provides examples of sources:

- Amounts (entered, accounted, and conversion data) and dates
- Descriptions
- accounts
- Sequencing information
- Third party details

Study the accounting transaction objects data model used in the Financial Services Accounting Hub. The data model provides detailed information on the different types of transaction objects used by the application. Accounting transaction objects are the views and tables that store transaction data in the standardized form required by the accounting program.

Data stored in transaction objects must also satisfy accounting transaction objects validation rules. These rules concern both completion and validation of the data.

See: Introduction to Transaction Objects, page 3-1

3. Analyze and Map Current Subledger Journal Entries

Complete an analysis of the journal entries currently generated by their applications and then compare this information to the options provided in the AMB. This exercise determines what AMB components must be seeded for their applications to enable the creation of those journal entries.

Seeded components include the journal entry descriptions, account derivation rules, and journal line types. These AMB components are referred to as subledger journal entry setup components. The objective is to make the upgrade to the Financial Services Accounting Hub transparent to users.

Such an analysis should, at a minimum, answer the following questions for the subledger journal entries created by the product:

- Under what conditions are each of the lines in the subledger journal entry created?
- Do the journal lines have actual, encumbrance, or budget balances?
- What is the side, debit or credit, of each subledger journal entry line?
- What description is used for the subledger journal entry?

- How are the accounts derived for the entry?

This list is not comprehensive. Ensure that all elements of current accounting are analyzed and mapped to AMB components and seed data.

Definition and Build

4. Seed Events Information in AMB

Once the events are determined, event information is seeded into the AMB. The AMB is used to create application accounting definitions.

> **Note:** Before seeding events information or sources in the AMB, register the applications with the Financial Services Accounting Hub.
>
> See: Registering Subledger Applications, page 4-7

Each accounting event should be represented by an accounting event type. These types are registered in the AMB. When subledger journal entries need to be created, the event type determines which application accounting definitions should be used to process the accounting event. Application accounting definitions created in the AMB determine the lines, descriptions, accounts, and other elements of subledger journal entries.

You can also group event types that allows for the sharing of journal entry setups and source assignments for a collection of related event types. Group event types as event classes and event entities.

Event Classes (intermediate level of grouping)

Group accounting event types into user-orientated transaction categories called event classes. For example, group the event types Invoice Approved, Invoice Adjusted, and Invoice Canceled into the event class Invoices.

Then assign AMB components, such as journal line types, by event class within the application accounting definition. This assignment simplifies setup when the accounting requirements for all event types in a class are the same. Also, sources assigned to an event class are available for the accounting of all event types in that event class.

Event Entities (highest level of grouping)

Group event classes into technical transaction models called event entities. For example, group the event classes Invoices and Prepayments into the event entity Invoices because both classes of transaction are stored in the Payables invoice transaction table (AP_INVOICES_ALL). Event entities enable you to treat events for a single transaction model in the same way. The event entity often logically corresponds to a single document used as a basis for several related transactions.

The following windows in the AMB relate to the seeding of event information:

- Entities
- Event Classes and Types
- Process Categories
- Event Class Predecessors

See: Seeding Event Information Using Accounting Methods Builder (AMB) Introduction, page 4-1

5. Code Calls to the Financial Services Accounting Hub Event Capture Routines

Using APIs provided by the Financial Services Accounting Hub, create programs to capture their accounting events. The accounting program uses the events to create subledger journal entries.

The Financial Services Accounting Hub provides the following APIs for creating and updating accounting events:

- Get Event Information APIs to get event information related to a document or a specific event
- Create Event APIs to create accounting events, individually or in bulk
- Update Event APIs to update events and keep them consistent with related transaction data
- Delete Event APIs to delete events

See: Seeding Calls to Subledger Event Capture Routines Introduction, page 8-1

6. Build Transaction Objects and Programs

The transaction objects process captures source data for accounting events and stores them in transaction objects. To successfully process the accounting event, the accounting program selects the source values it requires from these transaction objects.

Implementers are responsible for building and supporting the transaction objects. Complete the following tasks to accurately secure and prepare the transaction objects data:

- Create transaction objects that are used to store the transaction objects data.
- Write programs that populate the transaction objects with source values for each accounting event.

> **Note:** First register all transaction objects in the AMB.

There are four types of accounting transaction objects as listed below. The differences in these objects relate to whether they are used for header, line level, or ledger currency sources and whether they hold translated values. The accounting transaction objects are as follows:

- Transaction Objects Headers for untranslated header level sources
- Transaction Objects Headers MLS for translated header level sources
- Transaction Objects Lines for untranslated line level source values
- Transaction Objects Lines MLS for translated line level sources

Transaction Objects table or view names are accounting event class specific. The Create and Assign Sources program uses the transaction objects column names to generate source codes.

It is possible to define multiple tables or views for a single transaction object type and event class. For example, define two transaction objects line tables or views for the event class Invoices.

It is also possible for event classes to share transaction objects. For example, use the same transaction objects line table or view for the event class Invoices and the event class Credit Memos.

See:

- Types of Sources, page 3-6
- Overview of Building Your Database Objects, *Oracle Applications Developer's Guide*
- Overview of Using PL/SQL in Oracle Applications, *Oracle Applications Developer's Guide*

7. Run the Create and Assign Sources Program to Seed Sources in the Financial Services Accounting Hub

The Create and Assign Sources program automatically generates sources and source assignments based on the transaction objects definitions that are established in the previous step. (The transaction objects column names are used to generate sources.) These sources are available to create journal entries for transactions in the subledger and are part of the startup data for the Financial Services Accounting Hub. By seeding sources and source assignments, the program ensures a consistency between accounting transaction objects and the source and source assignments.

The Create and Assign Sources program also validates the transaction objects and journal entry setups as follows:

- The program verifies that each transaction object exists for an event class and that it is valid.

- If sources or source assignments exist, the program validates that the definitions are consistent with the transaction objects.
- If accounting attributes are mapped, the program validates that these mappings are consistent with the accounting attribute definitions.

See: Create Sources and Assignments, page 3-14

8. Revise Source Definitions and Assign Accounting Attributes

Once the Create and Assign Sources program has completed successfully, revise the source definitions before they can be used in the Financial Services Accounting Hub. These revisions include the following:

- Source names should be consistent with the Financial Services Accounting Hub attribute naming standards.
- Sources that correspond to Accounting Flexfield identifiers should are marked as Accounting Flexfield.
- Whenever appropriate, sources should have lookup types or value sets assigned.
- Sources that implementers anticipate will be used frequently in journal entry setups should be marked as displayed.

Manual Creation and Assignment of Sources

If users choose to manually create sources instead of running the Create and Assign Source program, then a source definition includes manually assigning each source to event classes. Assigning sources to an event class makes them available for creating application accounting definitions using that class.

Assigning Sources

Once standard sources are assigned to event classes, you may need to assign them to accounting attributes. Accounting attributes are a special category of standard sources. If an accounting attribute is relevant to the accounting event's underlying transaction, then its value is required to generate subledger journal entries for the transaction. Examples of accounting attributes are Entered Currency Code and Entered Amount.

Use the following windows in the AMB to seed and assign sources and accounting attributes:

- Sources
- Source Assignments
- Accounting Attribute Assignments

See: Revise Source Definitions and Assign Accounting Attributes Introduction, page 6-1

Accounting Program Integration

9. Integrate Subledger Applications with the Accounting Program

To enable the implementation of the Financial Services Accounting Hub, integrate the Create Accounting program into their applications. For Oracle Applications, this integration is made using APIs. For customizations, the Financial Services Accounting Hub provides workflow business events.

In addition, there are some changes that need to be undertaken to application windows. Some of these changes are as follows:

- Change application windows to accommodate the Financial Services Accounting Hub solution.

  This includes changing application windows to call the Create Accounting program.

- Remove obsolete windows and menu options.

  For example, options to transfer journal entries to the General Ledger become obsolete upon the uptake of the Financial Services Accounting Hub.

- Add the Financial Services Accounting Hub menu options.

  The Financial Services Accounting Hub provides seeded menus for users and implementers.

  The menu options should be added to seeded application responsibilities as appropriate.

See: Accounting Program Workflow Business Events for Non-Oracle Ledgers, page 9-1

Implement and Test

An application accounting definition is a grouping mechanism in the Financial Services Accounting Hub used to assemble a consistent set of application accounting definitions for an application. The application accounting definitions determine the accounting for an application's events.

An Application accounting definition includes assignment of the following components:

- journal entry descriptions (both header and line)
- journal line types
- account derivation rules

- journal lines definitions
- supporting references

All of these components along with the application accounting definition are created in the AMB. To users without custom requirements, the upgrade to the Financial Services Accounting Hub should be transparent. Therefore, application accounting definitions that replicate the pre-Financial Services Accounting Hub uptake journal entries, need to be seeded in the AMB.

The following sections describe the tasks that must be completed to create application accounting definitions.

10. Seed Application Accounting Definitions in the AMB

Seed default subledger journal entry setup components in the AMB. These setups are assigned to startup application accounting definitions.

The startup application accounting definitions should enable users to create subledger journal entries that are equivalent to the accounting generated before the uptake of the Financial Services Accounting Hub. For implementers, the goal should be to at least replicate the current default subledger journal entries. However, users can choose to enhance a subledger journal entry using any of the features provided by the AMB.

You can attach conditions to many of these components. A condition combines constants, source values, and operands and indicates when a particular journal line type, description, or account derivation rule should be used based on the source values.

All subledger journal entry setups are assigned to an accounting event class or type to determine how subledger journal entries for that class or type should be created.

Once the seed data is created, users must set up at least one application accounting definition for every subledger application.

See: Application Accounting Definitions, *Subledger Accounting Implementation Guide*

This section describes the components that must be seeded for subledger journal entry setup.

Journal Line Types

Journal line types determine basic information about a subledger journal entry line. Such information includes whether the line is a debit or credit, whether it should be transferred to the GL in summary or detail mode, whether matching lines should be merged, and its balance type (Actual, Encumbrance, or Budget).

Journal Entry Descriptions

Descriptions are included on subledger journal entry headers and lines. Include constant and source values in descriptions.

Account Derivation Rules

Account derivation rules determine which account should be used for a subledger journal entry line.

Journal Lines Definitions

Journal lines definitions enable users to group and assign journal line types, account derivation rules, and journal entry descriptions into a complete set of journal entry setups for an event class or event type. These sets can be shared across application accounting definitions for the same application.

Supporting References

Supporting references can be used to optionally store additional source information about a subledger journal entry and are useful for accounting reconciliation and analysis.

11. Test the Uptake of the Financial Services Accounting Hub.

Once the setup is complete, testing should be comprehensive to ensure that all accounting is correctly generated by the Financial Services Accounting Hub. Testing by implementers must incorporate accounting events, transaction objects data, and seed data. This should, at a minimum, include the following:

- Test that accounting events are successfully created.

  In addition, an accounting event must only be created when appropriate.

- Test that the sources seeded in the AMB are available for creating subledger journal entries.

  This should include testing sources that are not included in the default or seeded application accounting definitions.

- Test that subledger journal entries are successfully created for accounting events.

  This includes, where appropriate, testing of reversing, cross currency, and deferred subledger journal entries.

- Test that subledger journal entries contain the appropriate dates, amounts, descriptions, and accounts.

  If conditions are used to determine journal line types, account derivation rules, or descriptions, test these conditions as well.

- Test that all the subledger journal entries and lines are of the appropriate balance type (Budget, Actual, or Encumbrance).

- Test that all the seeded application accounting definitions can be successfully copied and modified by users.

- Test that all entries are successfully summarized when transferred to General Ledger if the summarization option is selected.
- Test that all the subledger journal entries can be successfully transferred and posted to General Ledger.
- Test that the accounting program creates subledger journal entries, transfers them, and posts them to General Ledger.
- Test that it is possible to successfully run inquiries on subledger journal entries using the Financial Services Accounting Hub inquiry functionality.
- Test that upgrade strategies are successful.

The above list is not intended to be comprehensive. Determine and complete appropriate testing before releasing the Financial Services Accounting Hub for your application.

The following points on security and upgrading security must also be taken into consideration when implementing the uptake of the Financial Services Accounting Hub.

Security

Incorporate event security guidelines when integrating with the Financial Services Accounting Hub.

As the implementation of the uptake to the Financial Services Accounting Hub is generic for all applications, the architecture provides a common solution to enforce event security. Event security refers to the way in which subledger accounting events are secured; only events to which users have access, under the subledger's security mechanism, are eligible to be processed.

The Financial Services Accounting Hub secures events by storing the security context of the transaction to which the event belongs. As an example, consider a transaction secured by operating unit. When events are created for the transaction, subledgers provide the operating unit data as well. This information is stamped on the event representing that transaction. Events inherit the security of their source transactions.

Determine the security policies needed to establish security on accounting events. Guidelines are provided to help you set up your application security when integrating with the Financial Services Accounting Hub.

See: Financial Services Accounting Hub Security Introduction, page 11-1

Upgrade Strategy

Complete an upgrade strategy for your application. The strategy should address the following issues:

- The cutover mechanism for current customers

  This ensures that all transactions completed before the uptake of the Financial Services Accounting Hub are accounted. Otherwise, it might be necessary to create events during the upgrade to account for transactions that are completed before the uptake.

- The need to modify or remove accounting options that are obsolete due to the uptake of the Financial Services Accounting Hub.

  Complete an analysis of current accounting options to determine which ones to make obsolete. Then devise a strategy for eliminating or migrating these options to the Financial Services Accounting Hub.

- Changing or eliminating accounting based subledger reports

  Since applications no longer store journal lines, any reports based upon such lines must be examined.

  Consider whether the Financial Services Accounting Hub reports fulfill their requirements, thereby making current reports obsolete.

- Removal or modification of journal entry inquiries

  The Financial Services Accounting Hub includes inquiries on both accounting events and subledger journal entries. However, Financial Services Accounting Hub inquiries can only be used to inquire on entries created by the Financial Services Accounting Hub.

Roles in the Uptake and Use of the Financial Services Accounting Hub

Although the term implementer is used throughout this guide, multiple roles are involved in the uptake and use of the Financial Services Accounting Hub. Whereas there may be some overlap of the tasks described in this chapter, the primary roles are as follows:

- Application Analyst, page 1-20
- Implementer, page 1-20
- Users, page 1-20

> **Note:** All roles should take part in testing to ensure that accounting is correctly generated by the Financial Services Accounting Hub.

See: Summary Listing of Uptake Steps, page 1-7

Application Analyst

The application analyst first undertakes a complete analysis to determine what events to capture. This analysis results in identifying all relevant events and their attributes. In addition, an analysis of events establishes the information to set up event entities, event classes, and event types.

Application analysts also analyze transaction objects requirements to determine what information (source data) is necessary to successfully create subledger journal entries from transactions. All sources that are required to complete the journal entry must be identified.

Finally, application analysts analyze and create application accounting definitions for a subledger.

See: Application Accounting Definitions, *Oracle Subledger Accounting Implementation Guide*

Implementer

Using event information provided by application analysts, implementers perform the following tasks:

- Seed event information like event entities, classes and types in the AMB
- Code the API calls to event capture routines
- Define transaction objects to store the source values
- Write programs that populate the transaction objects with source values for each accounting event
- Run the Create and Assign Sources program to define and assign sources

  These sources are available to application accounting definitions that create subledger journal entries. Before using them, certain source attributes must be revised. They must also be assigned to accounting attributes in the AMB.
- Integrate their applications with the accounting program

User

User refers to the following:

- Subledger accounting implementers who seed the appropriate application accounting definitions in the AMB to enable the creation of journal entries

  These definitions include those for journal entry descriptions, account derivation rules, journal line types, journal lines definitions, and supporting references.

- Implementers who define and modify application accounting definitions on an ongoing basis as required by the business environment
- Users who submit the accounting program to create subledger journal entries for transactions that have taken place

Secondary or Reporting Ledger Historical Upgrade

Using this functionality you can upgrade your secondary and reporting ledger data into Financial Services Accounting Hub.

If you are a:

- New Oracle Applications Release 12 customers and have a Secondary or Reporting ledger to an existing primary ledger that has open transactions
- Existing customer who upgraded from previous releases of Oracle Applications and then added Secondary or Reporting ledger to the Primary ledger

This utility lets you create Secondary or Reporting ledger representations for the newly added Secondary or Reporting ledger in XLA tables. You can choose the historical starting period for converting the transactions. Secondary or Reporting ledger representations are created based on the Primary ledger data by using the exchange rate on the initialization date in the setup. All the historical transactions are converted based on single exchange date so there are no exchange gain or loss.

Prerequisites

The following prerequisites apply to this feature:

- Exchange rate exists between all the transaction currencies and the reporting functional currency for the initialization rate date and type.
- All the SLA data for the primary ledger is balanced.
- Data in the reporting ledger may not be balanced (that is sum of the accounted debit may not be equal to sum of the accounted credit for a given header).
- Data is committed after all the records are inserted in the tables for any given period. If the volume of transaction is high, then sufficient rollback segment size must have been used.
- You cannot update the Enable ALC flag. If you add a Secondary or Reporting ledger and run the upgrade utility, then the entries are not replicated for that Secondary or Reporting ledger if the application is not ALC enabled in SLA.
- You cannot drilldown from GL as entries replicated are not transferred to GL.

- Secondary ledger conversion is not performed if the secondary ledger SLAM is different from the primary ledger. The secondary ledger only the currency, chart of accounts, or calendar may differ during conversion from the primary ledger.

The upgrade utility for SLA resolves the open, reversible, and future dated transaction conversion issues by converting the transactions to the Secondary or Reporting ledger. To maintain consistency and synchronization between the Secondary or Reporting ledger amounts in SLA and General Ledger, all transactions are converted using a daily rate based on the user-defined conversion rate date and conversion type. Run the utility for each reporting currency separately Convert all transactions using a daily rate based on the user defined conversion rate date and conversion type. The utility converts from the transaction currencies to the Secondary or Reporting ledger. If the transaction currency is same as the Secondary or Reporting ledger then the utility converts it using an exchange rate.

Fixed Number of Periods

This utility does not convert transactions that are prior to fixed number of periods even though they may not have completed the accounting life cycle. You have rerun the upgrade utility for those periods that may have open transactions.

Initial Balances for Supporting References

You can uploade initial balances for Supporting References (also known as Analytical Criteria). Initial balance is the starting balance for a supporting reference (Analytical criteria) in context of an application, ledger and CCID. There is no Journal Entry in SLA to support initial balances. Initial Balance is in the Ledger currency.

Defining Initial Balance for Supporting Reference Source Value

You define initial balances for a supporting reference source value combination for applications such as ledger and CCID. An initial balance is in the ledger currency for a period and contribute to the supporting reference source value combination balance calculated on any day going forward the date of initial balance.

The balance for supporting reference source value combination on any period prior to the date of initial balance will be non-existent.

You can define initial balances as Net of Debit and Credit balance. You can define initial balance for a closed period and even after the SAL has created balances and entries for supporting reference value.

Updating Initial Balances

Once the initial balance has been entered, you can modify the initial balance. It impacts balances in the periods following the initial balance.

To change the initial balance, upload a record for the same period, ledger, CCID for the supporting reference detail value combination for which the initial balance is uploaded. The program adds the delta (difference between the original uploaded value and changed value) to subsequent balance records.

Deleting Initial Balances

You can delete initial balances.

To insert or update Initial Balances for Balances by Supporting References:

- Upload data into the interface table.
- Submit concurrent program to Import data from interface table into the base table.
- View report generated from the concurrent program.

Upload Data to Interface Table

You can use the PL/SQL to load this interface table with control account initial balances from the legacy systems. Other EBS suite products, such as Oracle Lease Management can compute the initial balances for their seeded supporting references and populate this interface table.

Import Supporting References Initial Balances Concurrent Program

A concurrent program, Import Supporting Reference Initial Balances, is added to import data from interface table to the base table. This concurrent program is submitted as a request from SRS form.

Import Supporting Reference Initial Balances Using the XML Publisher Template

This feature includes a new template, Import Supporting Reference Initial Balances Report, to provide information about the Import Supporting Reference Initial Balances process. You can implement using this for data migration from Legacy systems for uploading balances.

2

Analysis of Accounting Events

Creating Subledger Journal Entries from Accounting Events

Accounting events have financial accounting significance and are used as a basis for the recording of financial information.

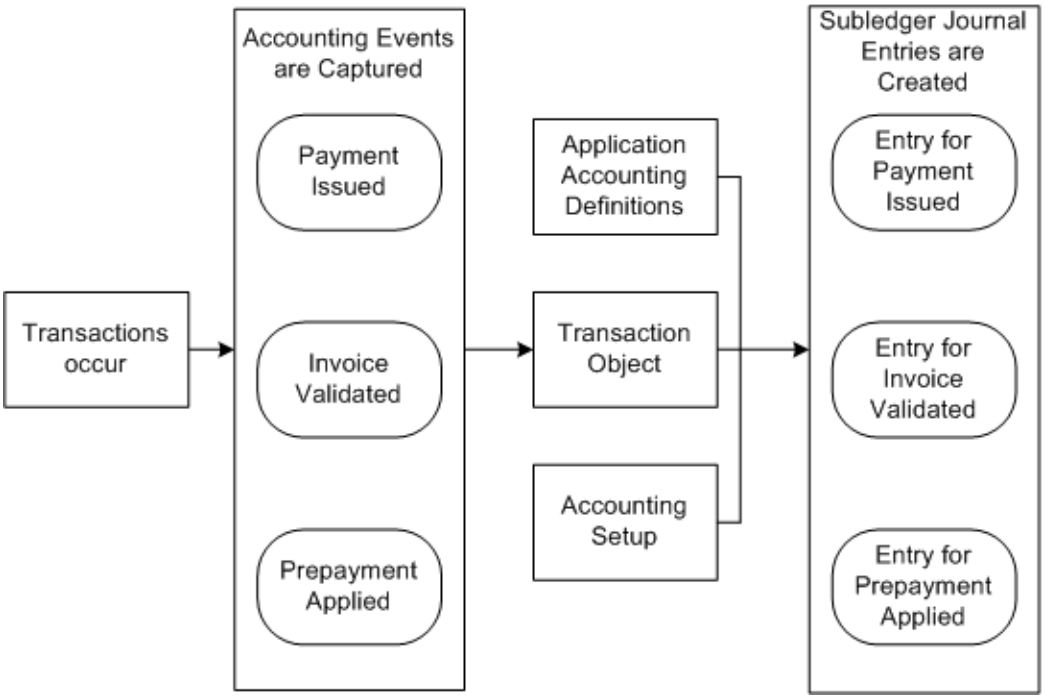
It is up to you to decide when an accounting event should be created. Accounting events can be created along with the entry or storage of transactions or their creation can be independent of transaction entry. For example, an import process can import documents first and then create accounting events as a separate step.

Capturing an event indicates that the event has occurred. Once events are captured, users can determine if and how they should be accounted.

See: Overview of the Financial Services Accounting Hub, page 1-4

The diagram below describes how to create subledger journal entries from the accounting events process using Oracle Payables as an example and is explained in the succeeding text.

Creating Subledger Journal Entries from Accounting Events Using Oracle Payables



As illustrated in the above diagram, transactions, such as Payment Issued, Invoice Validated, and Payment Applied, results in the capture of accounting events. When users signal that a transaction is ready to be accounted, the event is eligible for processing. For each eligible event, the accounting transaction objects passes contextual information about the event (source values) to the accounting program.

Financial Services Accounting Hub setups associated with a ledger use the source data for the event to create the appropriate subledger journal entry. Each accounting event can be used to create one or more subledger journal entries. Subsequently, the accounting event links transactions to their corresponding subledger journal entries.

The purpose of this chapter is to help identify the accounting events that must be captured. These events result in the creation of subledger journal entries.

Identifying Accounting Events

As a general rule, capture and record as accounting events all situations used to create General Ledger journal entries. The following procedures can assist in the analysis and identification of accounting events:

- Identify transactions that can impact your financial reports.

  Accounting events have financial significance for which financial information (accounting) can be recorded.

- Identify the life cycle of the transaction and the business events that change the state of the transaction throughout its life cycle.

- Identify distinct transaction objects data for each accounting event.

  This is necessary to record its full accounting implications.

See: Introduction to Transaction Objects, page 3-1

For maximum flexibility, all business events that the transaction model can reliably provide transaction amounts and contextual data necessary to create accounting for should be captured as accounting events. Such an approach allows greater flexibility in the creation of subledger journal entries.

> **Note:** There are special guidelines for handling changes in third party or third party site information for transactions that have already been accounted.
>
> See: Third Party Merge, page 2-20

The following sections provide examples of these procedures for identifying accounting events:

- The Financial Significance of Accounting Events, page 2-3

- Identifying the Life Cycle of the Transaction and Business Events that Change the State of the Transaction, page 2-4

- Ability to Record the Full Accounting Implication of Accounting Events, page 2-4

Financial Significance of Accounting Events

Business events continually occur within applications. Most of them have operational significance. These events range from the procurement of material and the hiring of employees to the manufacture of goods and collection of revenues.

Some business events require the recording of financial accounting information and are known as accounting events. The following examples have a financial accounting impact and therefore are accounting events:

- Applying a receipt to a Receivables line

- Approving or validating an invoice

Placing a payment hold on a Payables invoice changes the state of an invoice but it does not have a financial accounting impact and therefore, it is not an accounting event.

Not all users account for these events. As an example, consider an invoice in Payables. The invoice-validated event results in accounting if the accounting method is Accrual, but may not result in accounting if the accounting method is Cash Basis.

The capture of an accounting event is therefore completely independent from any subsequent accounting representation. All business events that can have an accounting impact should be captured as accounting events, irrespective of whether the events are eventually accounted.

Identifying the Life Cycle

The state of a transaction can change several times. Consider the following examples:

- A user selects the Complete option for a Receivables invoice. If the invoice is successfully completed, then its state is changed to completed.
- A user selects the Retire option for a fixed asset. Once the retirement is complete, the asset state is changed to retired.
- A user runs the Projects process to distribute the costs of an expenditure batch. After the process has successfully completed, the state of the batch is changed to distributed.
- A user enters a Payables invoice. The user then approves the invoice, but it goes on hold. The problem is resolved and the invoice is successfully approved. The state of the invoice is changed for both the hold and the approval.

In each of the above cases the state of the transaction is changed by the business event. There is also a corresponding financial accounting impact. As a result, in all the above examples, a case can be made for capturing appropriate accounting events when the transaction is saved.

Ability to Record the Full Accounting Implications of Accounting Events

To enable the creation of subledger journal entries, the data for each accounting event must be identified distinctly from the data for other accounting events. This also preserves an audit trail between the subledger journal entry and the transaction data for the accounting event.

Consider the business event, Invoice Validated. All the relevant information associated with this validation, such as the event date, invoice amount, invoice number, and distribution accounts, is unique. Therefore, the event qualifies as an accounting event. Transaction objects can capture and pass applicable data to the accounting program.

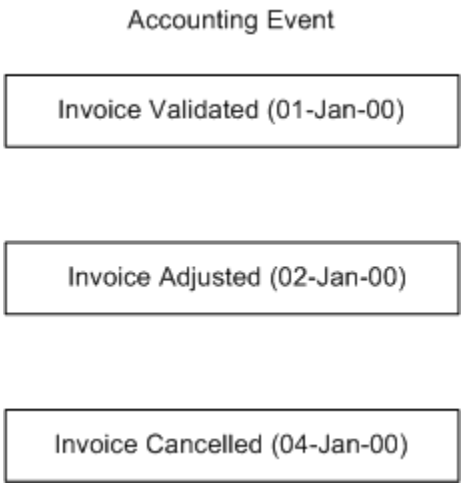
See:

See: Transaction Objects Introduction., page 3-3

Some transactions can have several accounting events. This is always true if the actions on a document have different event dates. Additionally, each accounting event can be a different accounting event type.

See: Event Type, page 2-7

As an example, consider a user working with a Payables invoice as described in the diagram below.



As the above diagram illustrates, the invoice is validated on 01-Jan-00. On the next day, an event date of 02-Jan-00, the invoice is adjusted. Two days later, 04-Jan-00, the invoice is canceled.

In this scenario, three accounting events, each with a different type, are used to track these actions. For each of these events, it is possible to derive the unique transaction data required to create accounting for the event.

Requirements for Valid Accounting Events

For an accounting event to be valid, ensure that it meets the following requirements:

- Must have a possible financial impact
- Must be related to a document or transaction
- Must contain links to its associated document or transaction
- Must have a valid status

Accounting Event Attributes

When an accounting event is captured, different event attributes are passed to the Financial Services Accounting Hub.

The table below lists some of the attributes that are stored for an accounting event along with their corresponding descriptions.

Accounting Event Attributes

Attribute	Description
Event ID	Accounting event internal identifier; provided by the accounting program
Event Number	Unique event sequence number within a document. The accounting program processes events in the order of their event number.
Event Type	For applications or transactions that are based on documents, event types typically correspond to the different operations that can be performed on the documents. For applications or transactions that are nondocument based, an event type corresponds to a particular kind of transaction.
Transaction Identifiers	Identifies the document or transaction in the subledger tables and constitute the primary key of the transaction
Event Status	Available statuses are Incomplete, Unprocessed, No Action, Processed.
Event Date	Date selected for processing. Each accounting event can have only one event date.
Transaction Context Values	Legal Entity, Ledger, and Asset Book
Application Specific Attributes	Additional columns are provided for implementers to store data drawn from the transaction model (state) at the time the event is captured. These can be useful in cases where the transaction data changes and information is needed on the original event.
Security context values	Provide the event's security context. Examples include organization identifier and asset book.

Attribute	Description
On Hold Flag	Indicates whether there is a gap ahead of an accounting event. If there is a gap, the event is put on hold. The Financial Services Accounting Hub does not process accounting events on hold due to a gap.

The Financial Services Accounting Hub event tables store the event data for these attributes. The presence of this data enables the creation of individual subledger journal entries for each accounting event. It also provides an audit trail between the subledger journal entry and the transaction data of the accounting event.

Event capture routines populate these tables when the events are created.

See: Seeding Calls to Subledger Event Capture Routines Introduction, page 8-1

The following sections describe selected accounting event parameters in more detail:

- Event Type, page 2-7
- Transaction Identifiers, page 2-7
- Event Status, page 2-10
- Event Date, page 2-16
- Transaction Context Values, page 2-16
- Security Context Values, page 2-17

Event Type

Event types categorize accounting events. Events with significantly different fiscal or operational implications are classified into different accounting event types. Accounting definitions in the Financial Services Accounting Hub are based on event types. An event type must be unique within an application, entity, and event class.

Transaction Identifiers

Transaction identifiers identify a subledger transaction associated with an accounting event. There are two types of transaction identifiers:

- System Transaction Identifiers, page 2-8
- User Transaction Identifiers , page 2-8

System Transaction Identifiers

System transaction identifiers provide a link between an accounting event and its associated transaction or document. For example, in Oracle Payables, CHECK_ID can link a Payment Issue event to the corresponding check (payment document). Support and technical personnel use these identifiers to trace and resolve errors.

System transaction identifiers constitute the primary key of the underlying subledger transaction. Typically, the identifier is the name of the surrogate key column on the transaction (header) associated with the accounting event. For example, the system transaction identifier for Payables invoices is the INVOICE_ID.

Whenever an accounting event is captured, the values of the system transaction identifiers are stored on the accounting event record along with other event data. Whenever system transaction identifiers are defined for the accounting event's event entity, the event capture mechanism validates that the system transaction identifiers are populated.

See: System Transaction Identifiers Window, page 4-14

User Transaction Identifiers

User transaction identifiers constitute the user-oriented key of the underlying subledger transaction. They are typically drawn from one or more database tables.

These identifiers are primarily used in accounting events inquiry and on accounting reports. Users use these identifiers to uniquely identify underlying transactions.

Example (Oracle Payables): User transaction identifiers for a Payables invoice include its invoice number, invoice date, supplier name, and supplier number.

Example (Oracle Receivables): User transaction identifiers for a Receivables invoice include its transaction type, transaction number, transaction date, customer name, and customer number.

The Financial Services Accounting Hub displays user transaction identifiers on accounting event reports and inquiries by dynamically generating SQL statements based on the definitions of both system and user transaction identifiers.

The following user transaction identifier characteristics are described in this section:

- Independent from the Transaction Model, page 2-8
- Independent from Accounting Event Status, page 2-9
- Up to Date, page 2-9

Independent from the Transaction Model

User transaction identifiers enable users to uniquely identify any subledger transaction. The transaction data that identifies a subledger transaction varies by event class. Accounting event reports and inquiries display the transaction identifiers and their

labels appropriate for the corresponding event class.

For example, the tables below list the transaction identifiers and their descriptions for an invoice and its corresponding payment. They are in two different event classes.

Payables Invoice

User Transaction Identifier	Description
Supplier Name	ABC Supplies
Supplier Number	1234
Invoice Number	Z-181723
Invoice Date	10-Jan-2005

Payment

Transaction Identifier	Description
Bank Name	<Your Bank>
Account Number	4567
Payment Method	Check
Payment Number	45641
Payment Date	20-Jan-2005

Independent from Accounting Event Status

User transaction identifiers can be displayed for an event regardless of its status. This includes the case where the accounting event has not been used to create subledger journal entries due to an error or the cases where it has not been processed.

Up to Date

User transaction identifier values are displayed at the time that the accounting event reports and inquiries are run. If a transaction identifier value has changed after the accounting event was captured, the accounting event reports and inquiries reflect the change.

Event Status

The event status is an indicator of what actions have been completed on the transaction and what operations are yet to be done. It is a snapshot of where the transaction is in its accounting event life cycle.

See: Event Status and the Accounting Event Life Cycle, page 2-12

Each event has an event status. The accounting program can make updates to this status. As an example, consider an accounting event that has a status of Unprocessed. Once the accounting program successfully processes the accounting event in final mode, the status of the event is updated to Processed.

See: Draft and Final Accounting, page 2-22

The table below lists the event statuses along with their corresponding details. At any point of time, an event can have only one of these statuses.

Event Status

Status	Details
Incomplete	The accounting event data is in the process of being created. Some of the accounting event data cannot be created at this point. There can be validations that have not yet been performed. No subledger journal entry exists for the accounting event. If the accounting program is run, it does not process accounting events with a status of Incomplete. The subledger application updates this event status. Example: A user enters a Payables invoice, but does not validate it. If an event is captured, it has a status of Incomplete.

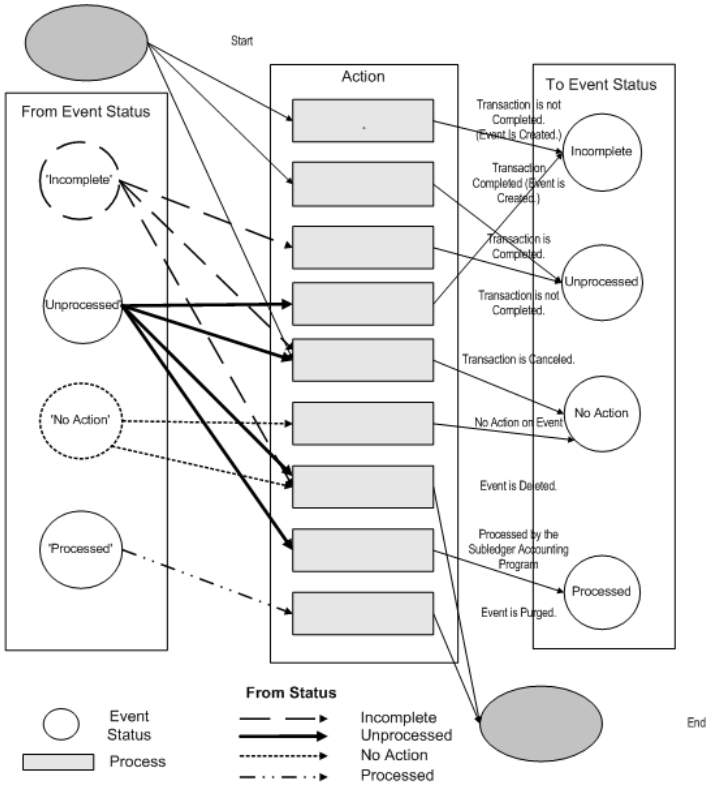
Status	Details
Unprocessed	All of the transaction data for this accounting event has been created and all validations have been performed. At this point of time, the subledger journal entry can be created. When the accounting program is run, it processes accounting events with a status of unprocessed. This event status is updated by the subledger application based on the transaction status. Example: A user successfully validates a Payables invoice. An accounting event is captured and has a status of Unprocessed.
No action	No subledger journal entry is needed for this accounting event. The subledger application updates this event status. Example: Assume an Invoice accounting event is created with the status Unprocessed. If the invoice is canceled on the same GL date, the Invoice accounting event should not be accounted. The event status is set to a status of No Action, since the invoice has no financial accounting impact.

Status	Details
Processed	All of the transaction data for this accounting event is created, all validations are performed and, if appropriate, the final subledger journal entry is created. The transaction data associated with the accounting event cannot be changed. For those transactions where multiple accounting events are allowed, any changes to the transaction data at this point of time results in a new accounting event. The changed transaction data is tracked under the new accounting event. After successfully creating subledger journal entries, the accounting program updates the event status. Example: Consider an Invoice accounting event which has a status Processed. Users cannot change the invoice currency at this point because it affects data potentially associated with an event that has already been accounted. However, you can enter new lines provided the subledger functionality allows such a change. New lines entered are recorded with new accounting events. See: Transactions with Multiple Events, page 2-23
Related Event in Error	If an event ends in status Invalid or Error, other events for the same document which are processed and accounted successfully in the same run of the Create Accounting program must be set to status Related Event in Error.

Event Status and the Accounting Event Life Cycle

Every event has a life cycle. The event status indicates what actions have been completed on a transaction and where the event is in its life cycle. The life cycle is shown in the diagram below.

Accounting Events Life Cycle



The above diagram illustrates all the status changes for an accounting event. The diagram has three blocks. The left block, From Event Status, shows the possible initial statuses of an event. These statuses are Incomplete, Unprocessed, No Action, and Processed.

The center block, Action, represents actions that users complete in subledger application, such as Oracle Projects, Oracle Assets, or Oracle Receivables. These actions result in events being created, processed, or deleted.

The right block, To Event Status, shows the possible final status based on the action in the central block. The status values include Incomplete, Unprocessed, No Action, and Processed Program.

Note that these are all the possible status changes. Not all accounting event types can support all of these status changes. For example, some accounting events, once they have a status of Unprocessed, cannot be updated to an Incomplete status. Implementers are responsible for determining the supported status changes for an event.

There may also be conditions that determine whether the accounting event can move from one status to another. These conditions can vary by accounting event.

Accounting Event Life Cycle Example

The following example of the event life cycle has two parts, both related to the same invoice. However, the events are different in each section. This is due to the following reasons:

- The invoice is regarded as completed in part one; accounting is created.
- The adjustment in part two relates to a different event date.

  Note that each accounting event can have only one event date.

Note that two accounting events are used if either one of these conditions are true; either the adjustment was for a different date or the invoice was already accounted.

Part 1. Creating and Modifying an Invoice

Accounting Event: Invoice Validated Date: 15-Jun-2005

Step 1. A user enters a purchase invoice. The table below lists the details of this invoice.

Line #	Amount	Account	Event Date	Accounted
1	$100	01-cc1-expacct1	15-Jun-2005	No
2	$100	01-cc2-expacct1	15-Jun-2005	No

An event is created upon creation of the invoice header. The distribution lines have the accounting status No. They remain in this status until the invoice is accounted in final mode. The event status is Incomplete.

Step 2. The user validates the invoice. As a result, the event status is updated to Unprocessed.

Step 3. The user adds a new distribution line. The event status is now Incomplete and the invoice must be revalidated.

Step 4. The user revalidates the invoice. As a result, the accounting event status is updated to Unprocessed.

Step 5. The user creates accounting for the accounting event in Final mode. As a result, the accounting and document status are both updated to Processed. The table below lists the details of the invoice. The lines now show a status of Yes for accounted.

Line #	Amount	Account	Event Date	Accounted
1	$100	01-cc1-expacct1	15-Jun-2005	Yes

Line #	Amount	Account	Event Date	Accounted
2	$100	01-cc3-expacct1	15-Jun-2005	Yes
3	$200	01-cc3-expacct1	15-Jun-2005	Yes

Part 2: Modifying an Invoice which is Accounted

Accounting Event: Invoice Adjusted Date: 16-Jun-2005

In Part 2, the invoice from Part 1 is modified. Because the invoice has already been accounted and the adjustment is for a different event date, a new accounting event is created to record the adjustment.

Step 1. The user adds a line to the invoice. This addition appears as line 4 in the table below, which lists the details of the invoice.

Line #	Amount	Account	Event Date	Accounted
1	$100	01-cc1-expacct1	15-Jun-2005	Yes
2	$100	01-cc2-expacct1	15-Jun-2005	Yes
3	$200	01-cc3-expacct1	15-Jun-2005	Yes
4	$50	01-cc4-expacct1	16-Jun-2005	No

An event is created to account for this adjustment. The event status is Incomplete.

Step 2. The user approves the invoice. As a result, the event status is updated to Unprocessed.

Step 3. The user creates accounting for the invoice. The new line 4 now has a status of Yes for accounted and the event status is updated to Processed. The table below lists the details of the invoice.

Line #	Amount	Account	GL Date	Accounted
1	$100	01-cc1-expacct1	15-Jun-2005	Yes
2	$100	01-cc2-expacct1	15-Jun-2005	Yes

Line #	Amount	Account	GL Date	Accounted
3	$200	01-cc3-expacct1	15-Jun-2005	Yes
4	$50	01-cc4-expacct1	16-Jun-2005	

Event Date

Each accounting event must have one and only one event date. If multiple GL dates are entered for a document, then one accounting event must be created for each GL date. For example, assume a user enters an invoice with two distributions, each with a different event date. In this case, two accounting events must be created, one for each event date.

As another example, assume a Payables invoice is approved and accounted. On the next event date, it is canceled. This creates two accounting events.

If the transaction has the potential to create multiple events, then both the event date and status of the previous event determines whether a new accounting event is created. Consider the following examples:

- A Payables user cancels an invoice before accounting for the Invoice event.

  If the event date for the cancellation is the same as that of the accounting event created for the invoice, then the status of the Invoice accounting event is updated to No Action. No accounting event is created for the cancellation.

- A new accounting event with the same event date as the original invoice now reverses the allocations.

  The status of the original Invoice accounting event is set to No Action and no accounting event is created for the reversal.

- An invoice is canceled after it has been accounted.

  The accounting event created for the invoice is not affected by the cancellation. A new event for the cancellation is created and requires processing to create a subledger journal entry.

Transaction Context Values

For the accurate creation of subledger journal entries, an accounting event's transaction context must be passed to the Financial Services Accounting Hub. This context includes the following:

- Ledger (mandatory), page 2-17

- Valuation Method (optional), page 2-17
- Legal Entity (optional), page 2-17

Ledger (mandatory)

A ledger represents a set of accounting information for a legal or management entity. Each ledger is associated with a chart of accounts, calendar, and currency for which accounting information is recorded.

Valuation Method (optional)

In some cases, the rules for how accounting events should be accounted require that they be subject to differing monetary valuations (valuation methods) for different ledgers. The Financial Services Accounting Hub assumes that each subledger transaction and accounting event is for a single valuation method.

See: Valuation Methods and Financial Services Accounting Hub, page 2-24

Legal Entity (optional)

Accounting events inherit the legal entity from the transaction it is associated with. Each transaction entity is stamped with a single legal entity. A single business transaction that crosses a legal entity boundary, such as a cross legal entity payment, results in one accounting event for each legal entity involved in the transaction.

Security Context Values

Security can be enforced if the application implements a security policy. In this case, only accounting events for transactions to which the user has access based on the security policy can be processed by the accounting program. When users submit the accounting program, only accounting events with operating units to which users have access are eligible to be accounted.

For example, if an operating unit secures a subledger's transactions, implementers can implement security so that accounting events inherit the operating unit of the transactions.

Accounting Event Analysis - Additional Notes

This section includes the following:

- Accounting Event Order, page 2-18
- Third Party Merge, page 2-20
- Audit Trail and Business Event Restrictions, page 2-22

- Draft and Final Accounting, page 2-22
- Transactions with Multiple Events, page 2-23
- Valuation Methods and the Financial Services Accounting Hub, page 2-24

Accounting Event Order

All events have an associated event type. As noted earlier in this chapter, for transactions that are document-based, event types correspond to the different operations that can be performed on a document. For transactions that are nondocument-based, an event type corresponds to a particular kind of transaction.

See: Event Types, page 4-18

If there is an implied order in the accounting events related to the different event types of a particular document, then implementers are responsible for enforcing that order. For example, an Invoice Validated event should have a lower event number and the same or an earlier event date than Invoice Adjusted events for the same invoice. The Financial Services Accounting Hub is neither aware of, nor does it enforce any implied order.

The accounting program processes accounting events according to criteria specified by users. All events whose dates are equal to or less than the end date specified when the accounting program is submitted are selected for processing.

Example: Implied Order in Oracle Payables

Assume the following three accounting event types for an invoice: Invoice Validated, Invoice Adjusted, and Invoice Canceled.

It should not be possible to generate a subledger journal entry to account for an invoice adjustment before the invoice validated event is processed. Similarly, it should not be possible to adjust the invoice after it has been canceled. Therefore there is an implied order to these event types.

Correctly Enforcing Implied Order

Assume that Payables has four accounting events for a single invoice as described in the table below.

Event Number	Event Type	Event Date
1	Invoice Validated	August 1, 2005
2	Invoice Adjusted	August 15, 2005

Event Number	Event Type	Event Date
3	Invoice Adjusted	August 16, 2005
4	Invoice Canceled	August 28, 2005

In this case, Payables has correctly ordered the accounting events. For example, since event processing is by event number, the Financial Services Accounting Hub does not attempt to account for the adjustment or cancellation of the invoice before the invoice approved accounting event has been processed.

The Invoice Canceled event, must receive an event number greater than the event number of the last Invoice Adjusted event type. Similarly, the Invoice Adjusted events receive event numbers greater than the Invoice Validated event but less than the Invoice Canceled event.

No Implied Order Between Events of the Same Type

In the following example, assume that event number 3 has an earlier date than event number 2 as described in the table below. Note that event number 2 and event number 3 have the same type of Invoice Adjusted.

Event Number	Event Type	Event Date
1	Invoice Validated	August 1, 2005
2	Invoice Adjusted	August 16, 2005
3	Invoice Adjusted	August 7, 2005
4	Invoice Canceled	August 28, 2005

If the user submitted the accounting program with an end date of August 10th, event 1 and then event 3 are processed. Events 2 and 4 remain unprocessed. The latter two events are selected during a later submission of the accounting program with an End date greater than August 28th. Since there is no implied order between the two Invoice Adjusted accounting events, this is acceptable ordering.

If the user submitted the accounting program with a To Date of the August 31st, all events are processed in event number order, with the August 7th adjustment being processed after the August 16th adjustment. Therefore, the order that the events are processed is affected by the To Date when submitting the accounting program.

Not Enforcing the Implied Order, Incorrect Dating of Events

The table below describes this example.

Event Number	Event Type	Event Date
1	Invoice Validated	August 1, 2005
2	Invoice Adjusted	August 15, 2005
3	Invoice Adjusted	August 16, 2005
4	Invoice Canceled	July 20, 2005

In this example, the events are appropriately numbered. If the user specified an end date equal to or later than August 16th, the events would still be processed in the correct order.

However, the events are not appropriately dated. If the user specified an end date less than August 16th, the accounting program would be unaware of the Invoice Adjusted event number 3, and the invoice canceled event on July 20th would be processed before the adjusted event.

In this example, Payables has incorrectly ordered the accounting events using the event dates. It is possible to account for the cancellation of an invoice before the invoice validated accounting event has been processed.

Third Party Merge

Third party merge is a business event that results in the need to update third party information on journal entries or impact third party control account balances. Situations that may require third party updates are described as follows:

- A company acquires another company
- Clerical input error
- A duplicate or erroneous third party is entered into the system

The third party merge event differs from other accounting events in the following ways:

- It involves many different transactions whereas other accounting events are related to a single transaction entity.
- It can affect many applications because it is not limited to one subledger.

  For example, when two companies merge, all related subledgers are affected, such

as Receivables, Payables, and Purchasing. The Financial Services Accounting Hub performs the accounting for all the subledgers involved.

- The accounting impact of third party merge results in one of the following:
  - Updating third party information on the existing journal entries
  - Transferring the third party control account balance and reversing the journal entries followed by a rebooking of the existing journal entries
- The accounting generated by third party merge is based on existing journal entries and not on transaction data.

Third Party Merge Process

The following components make up the third party merge process flow:

1. Third Party Merge Accounting option

   For every ledger and subledger application, users define this accounting option in the Update Accounting Options window in the Accounting Setup Manager.

   See: Financial Services Accounting Hub Options Setup Overview, *Oracle Subledger Accounting Implementation Guide*

2. Creation of the third party merge event

   When users record a third party merge operation, the corresponding subledger application calls the Third Party Merge program to create the event. The event is created with the following characteristics:

   - The event is attached to a seeded transaction entity.
   - The event is assigned a hard-coded event type.
   - The old third party and the new third party information is populated in reference columns.

3. Creation of journal entries

   The journal entries are created in one of the following ways:

   - Synchronously

     The third party merge is executed as soon as the event is created if this feature is implemented in the subledger application.

   - Asynchronously

     Users submit the Create Accounting for the Third Party Merge program to process the third party merge event.

The Create Accounting for the Third Party Merge program performs the following tasks for third parties:

- Identifies the existing journal entries that are impacted by the third party merge operation
- Depending on the third party merge accounting option selected, performs one of the following:
  - Updates the third party information in the existing journal entries
  - Transfers the third party control account balance as of the merge date and reverses and rebooks the journal entries after the merge date with the new third party information

    For the new journal entries just created, populates the description of the journal entry header and lines with the old and new third party information.
- Maintains a link between the original journal entry and the reversed journal entry
- Maintains a link between the new journal entries and their corresponding distributions

Audit Trail and Business Event Restrictions

Accounting event parameters enable an audit trail between the subledger journal entry and the transaction data for the accounting event.

Some business events have the potential to invalidate the accounting created by a previous accounting event. These business events should not be supported. Each event and its accounting should be auditable and reportable at a later point in time. As an example, it should not be possible to delete an invoice after the Invoice Validated accounting event is processed in final mode.

Users can change the transaction data if its associated event is unprocessed, but no changes should be allowed after the event is processed. As a general guideline, transactions should not support business events that call for a delete or update of accounting created by other accounting events. The business events should be changed to achieve the same results by reversing accounting created by the previous accounting events.

Draft and Final Accounting

Subledger journal entries can be created in draft or final mode. Both kinds of entries are created using the same accounting definitions, with the difference that it is possible to modify or delete transaction information associated with draft entries. Any changes to the transaction data after accounting in draft mode invalidates the draft journal entries.

In both draft and final mode, if errors are raised in processing of events for a

transaction, the accounting program does not endeavor to fully account that transaction. Instead those events are flagged and the program proceeds to the next transaction.

Draft journal entries are purged once the final accounting is created. Also, if there is any change in event status, event date, or event type of an event, then draft journal entries associated with this event are deleted. Draft entries are not transferred to General Ledger.

The table below summarizes the modes of accounting for events.

Accounting Issue	Draft Mode	Final Mode
Are journal entries created for the events?	Yes	Yes
Are subsequent changes to transaction data permitted?	Yes	No (enforced by the application implementers and not by the Financial Services Accounting Hub)

Transactions with Multiple Events

Guidelines for creating accounting events for transactions and documents with multiple events are as follows:

- If a previous event with the same GL date has a status of Unprocessed, merge accounting events within the same document.

  There should be at most one accounting event for the same document and same GL date that does not have the status Processed. One accounting event can include the data for multiple transactions. This enables accounting at one time for all transaction changes of the same accounting event type.

  As an example, an invoice which has not been accounted and therefore has an accounting event with the status of Unprocessed, is adjusted by adding a new line using the same GL date. The invoice is revalidated. Both the original and new lines must be included on the same accounting event.

- If the previous events for a document have a status of Processed, a new accounting event should be created.

  As an example, consider an invoice which has been previously approved and accounted. The user then cancels the invoice. A new accounting event must be created to record the cancellation.

Valuation Methods and the Financial Services Accounting Hub

In most applications there is one transaction recorded for each underlying accounting event. For example, when an invoice is issued to a customer or received from a supplier, a single invoice transaction is registered in the system. Although there may be many possible interpretations of the financial impact of the accounting event, and therefore many possible accounting representations of it, the underlying event has only one monetary value in the entered currency. Therefore, only one transaction is captured in the corresponding subledger application.

However, in some applications, a single underlying business event can give rise to one or more transactions, each with a different monetary value. For example, a fixed asset can be depreciated and therefore valued according to corporate policies in a corporate book and according to tax rules in one or more tax books. Each of these transactions may require accounting.

In the Financial Services Accounting Hub, each valuation of an underlying accounting event is called a valuation method. A valuation method can be considered to be a method of accounting for events in a particular environment. It is uniquely identified by a valuation method identifier. The Financial Services Accounting Hub supports a different accounting for each valuation.

Oracle E-Business Suite examples: In Oracle Assets, each asset book represents a valuation method. Another example of a valuation method is the costing method used by Oracle Cost Management and OPM.

Valuation Method and Accounting Representations

A ledger's subledger accounting method, accounting currency, calendar, and chart of accounts determine the nature of the accounting representation. Typically, an accounting representation refers to all journal entries that are stored in a ledger.

See: Accounting Representations, page 1-2

Like accounting representations, each valuation method often corresponds to a fiscal or corporate valuation of the underlying accounting events. In Assets, for example, the corporate book is often considered to be the corporate representation and the tax book the fiscal representation.

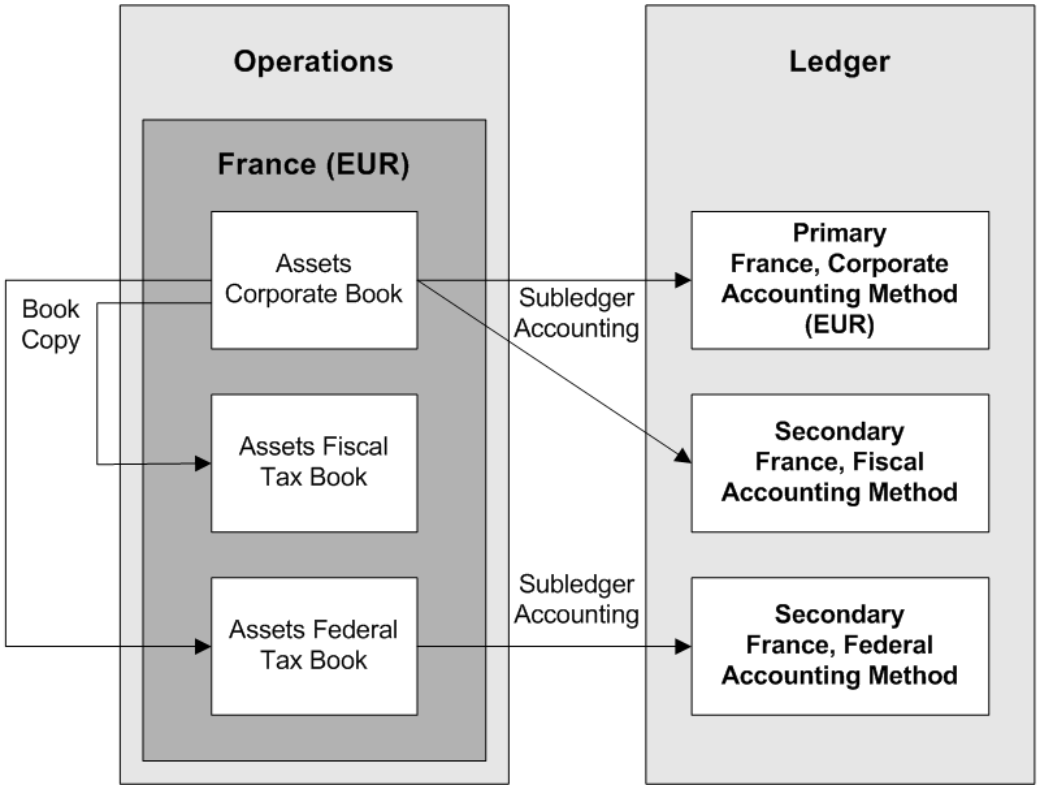
This aspect of valuation methods has important implications for the Financial Services Accounting Hub because each valuation method is in itself the equivalent of an accounting representation. For example, for an enterprise that has adopted a corporate first accounting standard, the corporate asset book feeds the primary (corporate) ledger and the tax asset book feeds the secondary (fiscal) ledger.

The table below summarizes the differences between standard applications in the Oracle E-Business Suite and those that support multiple valuation methods.

Standard Application	Valuation Method Application
A subledger transaction is associated with the primary ledger only.	A subledger transaction is associated with either the primary ledger or a secondary ledger.
One accounting event is created per transaction. Accounting is simultaneously generated in all secondary ledgers.	Multiple transactions are created per business event, one for each valuation method, each of which are accounted.

The difference between valuation and nonvaluation method applications is that valuation method applications link their representations directly to their respective primary and secondary ledgers. In nonvaluation method applications, the representations are only linked to the primary ledger.

Consider the following example, illustrated in the figure below, to demonstrate the differences between the two approaches.



In the diagram above, the Operations box for a French company shows a valuation method application Assets, which includes Assets Corporate Book, Assets Fiscal Tax Book, and Assets Federal Tax Book. The Ledger box shows the corporate fiscal, local fiscal, and federal accounting representations, ledgers for a French subsidiary of a

multinational corporation.

Applications that opt to enable the use of multiple valuation methods also have the flexibility to choose whether to use the valuation method of the primary accounting representation for a secondary accounting representation or an alternate valuation method. For example Assets can choose to create accounting for the fiscal accounting representation based on the corporate book transactions rather than the fiscal tax book transactions. Similarly, Assets can choose to create accounting for the federal accounting representation based on the federal tax book transactions rather than the corporate book transactions.

The diagram above shows that the accounting events for the corporate book create accounting for both the primary (Corporate Accounting Method) and secondary ledger (Fiscal Accounting Method) simultaneously. The accounting events for the Assets Federal Tax Book only create accounting for the secondary ledger (Federal Accounting Method). Note that transactions must not be created for the Assets Fiscal Tax Book because its associated secondary ledger (Fiscal Accounting Method) already receives its accounting from the accounting books of the corporate book.

The Financial Services Accounting Hub creates accounting for the ledger associated with the accounting event. For applications that support multiple valuation methods, the accounting events should be associated with the appropriate primary or secondary ledger. For applications that do not support multiple valuation methods, the associated ledger is always the primary ledger. Accounting is simultaneously created for all the secondary ledgers.

3

Transaction Objects Analysis

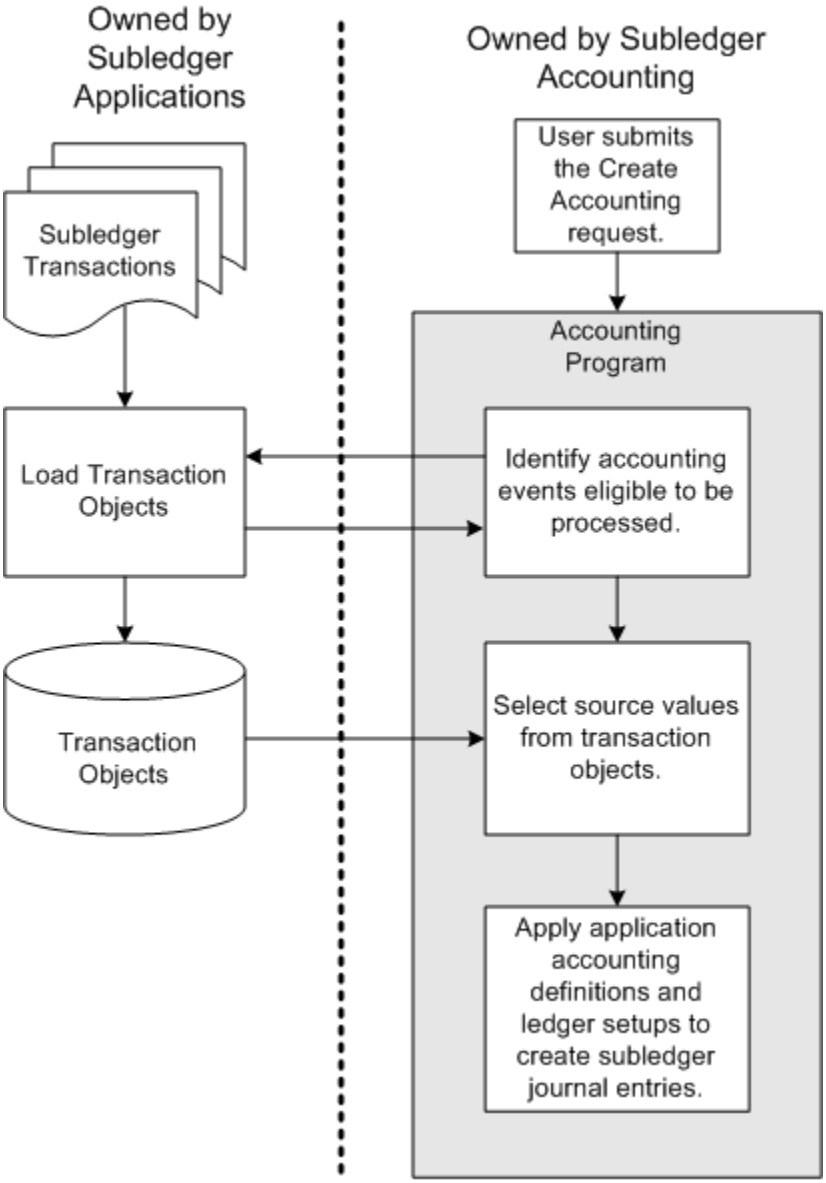
Transaction Objects Overview

For each eligible accounting event, source data from the transactions is stored in transaction objects (tables or views). To successfully complete the processing of accounting events, the Financial Services Accounting Hub selects the source values it needs from these transaction objects.

This chapter provides details on how the Financial Services Accounting Hub uses the transaction object data model to generate accounting.

The figure below provides a high-level overview of the transaction objects process and the Financial Services Accounting Hub. Implementers implementing the uptake of the accounting program are responsible for creating and populating the transaction objects.

Financial Services Accounting Hub Structure and Transaction Objects



The figure above illustrates the structure of the Financial Services Accounting Hub and its relationship with transaction objects. As shown, the Load Transaction Objects program is called by the accounting program for each accounting event eligible to be accounted. The Load Transaction Objects program populates the source values for the event into the transaction objects.

Transaction objects refer to the tables or views from which the Accounting Program takes the source values to create journal entries. The phrase, "stored in the transaction objects," means that the source values are available as column values in transaction objects to the accounting program. You can create a program to populate columns in

transaction objects or use views of your transaction data as the transaction objects. Transaction objects and load programs are created and maintained by subledger applications. The Financial Services Accounting Hub manages the interaction between the transaction objects and the accounting program.

The accounting program performs the following tasks:

- Identifies accounting events eligible to be processed
- Populates the events temporary table
- Raises the oracle.apps.xla.accounting.extract workflow business event, which can only be used for customizations

The list of accounting events eligible to be processed by the accounting program is stored in the accounting events temporary table (XLA_EVENTS_GT).

See: Accounting Program Workflow Business Events for Non-Oracle Ledgers, page 9-1

The accounting program selects the source values it requires from columns in the transaction objects. Using these source values, the accounting program applies application accounting definitions and ledger setups to process the accounting event and create subledger journal entries.

How Sources Are Used in the Financial Services Accounting Hub

See: Introduction to Events and Sources, page 1-3

Sources defined in the Accounting Methods Builder (AMB) are the basis of all application accounting definitions created by implementers and users. Use sources in setting up the following AMB components:

- Account Derivation Rules, page 3-3
- Mapping Sets, page 3-4
- Journal Entry Descriptions, page 3-4
- Third Party Control Accounts, page 3-4
- Supporting References, page 3-4
- Conditions, page 3-5

Account Derivation Rules

Use sources in account derivation rules to determine the Accounting Flexfields used in subledger journal entry lines.

An Accounting Flexfield or segment value stored on a transaction can be used as a

source. For example, users can create an account derivation rule that takes the Invoice Distribution Account source from a Payables invoice distribution and use it for an Expense subledger journal entry line.

Mapping Sets

Use mapping sets to derive Accounting Flexfield values from nonaccounting related sources. For example, if a source such as Asset Location is available, a mapping set can be based on it. This mapping could take the following form:

- Location A uses account segment value 1000.
- Location B uses account segment value 2000.

Journal Entry Descriptions

Use sources to determine what information is included in the descriptions of subledger journal entry headers and lines. For example, if a customer name is to be included in the description of a subledger journal entry, then it must be available as a source.

Third Party Control Accounts

Third party control accounts are natural accounts that record a subledger's balance or activity in relation to third parties. In many countries with detailed accounting regulations, legal requirements mandate the creation of views and reports of all subledger transactions and corresponding journal entry lines for a particular customer or supplier. Users can set up third party control accounts for each subledger application. Control account balances from subledger journal entries are maintained for each related third party.

> **Note:** The Financial Services Accounting Hub maintains third party balances for accounts that are flagged as third party control accounts.

See:

- Third Party Accounting Attributes, page 3-32
- Third Party Control Guideline, page 3-66
- Create Accounting Overview, *Oracle Subledger Accounting Implementation Guide*
- Defining Segment Qualifiers, *Oracle General Ledger User Guide*

Supporting References

Supporting references are used to store additional information about a subledger journal entry either at the header or line level. They can also be used to establish a subledger balance for a particular source and account. Users can decide on the

references they would like to use for a journal line type. Users use sources to set up supporting references.

By providing additional business information about the subledger journal entry and balances of an account, supporting references are useful for accounting reconciliation and analysis.

Conditions

Use conditions to determine which of several alternative journal line types, , or descriptions are used to create subledger journal entries.

Consider the following examples for using conditions:

- **Oracle Payables Example:** Users can elect to use the Distribution line type source to determine the subledger journal line type. For example, use the Freight journal line type if the Distribution line type is equal to Freight. The Distribution line type must therefore be defined as a source and be included in the transaction objects to allow users to create this condition.
- **Oracle Inventory Example:** Users may want to use a purchase price variance account for an inventory issue only when the Distribution line type is equal to Invoice Price Variance.
- **Oracle Receivables Example:** Users can elect to use a specific revenue account based on a transaction type.

GL Reconciliation Functionality

Specify a source to be used as a reconciliation reference with the GL Reconciliation feature. Reconciliation functionality enables reconciliation of transactions in any account that should balance to zero.

See: Reconciliation Reference Accounting Attribute, page 3-35

Transaction Objects Data Model

This section includes the following parts:

- Population of Transaction Objects, page 3-13
- Reference Objects, page 3-13
- Create Sources and Assignments, page 3-14

Types of Sources

The Financial Services Accounting Hub classifies sources as follows:

- Standard, System, and Custom Sources, page 3-6
- Untranslated and Translated Sources, page 3-7
- Header and Line Sources, page 3-7

Standard, System, and Custom Sources

Standard Sources

Standard sources are sources defined by the subledger application. They are stored in transaction objects and made available to the accounting program. The definition of what transaction information must be made available as standard sources vary by application and the kind of transaction.

Examples of standard sources for a Payables invoice are Supplier Type and Invoice Number. Examples of standard sources for a Receivables receipt are Bank Account Name and Receipt Number.

System Sources

Subledger accounting defines system sources and they are always available to the accounting program. Therefore, implementers do not need to define system sources or include them in their transaction objects. System sources include the following:

1. System variables, such as Today's Date or Language
2. Event and ledger sources

   These values are associated with the accounting event or with the ledgers for which the accounting event is being processed. Event Date is an example of an event source. Ledger Name and GL Period Name are examples of ledger sources.

   See: System Sources Guideline, page 3-74

Custom Sources

Users can use custom sources to extend the pool of sources available to create accounting definitions. Using standard and system source values as parameters, users can write PL/SQL functions that define custom sources.

See: Custom Sources, *Oracle Subledger Accounting Implementation Guide*

Untranslated and Translated Sources

Translated Sources

Some sources can be translated into more than one language. The translation can be created by users using Oracle MLS features or created as part of the seed data.

An example of a user-created translated source for the Costing module is Inventory Item Name. Inventory Item Name can be translated using MLS features.

An example of a seeded translated source for Receivables is Term Name. Payment Terms, such as Immediate, are delivered as seed data.

> **Note:** The Financial Services Accounting Hub offers features that help limit the number of values that are defined as translated sources.
>
> See: Translated Sources, Lookup Types and Value Sets Guideline, page 3-68

Untranslated Sources

Untranslated sources are source values that are never translated into more than one language. Examples of untranslated sources include all numbers, dates, and lookup codes.

Header and Line Sources

Header Sources

Header sources have the same value for all transaction lines or distributions associated with an accounting event. These sources are typically associated with a transaction header or with transaction reference data.

An example of a header standard source for a Payables invoice is Invoice Supplier Name. An invoice can have only one supplier name. This name is created at the header level and does not vary by line. Examples of header sources for a Receivables receipt application include Receipt Customer Name and Receipt Number. These sources have the same value for all related transaction lines.

Line Sources

Line sources have values that can vary by the transaction lines or distributions associated with an accounting event. They must be stored in the transaction objects at the line level.

Examples of line sources for a Payables invoice include Invoice Line Tax Code and Invoice Distribution Expense Account. Examples of line sources for a Receivables receipt application are Receipt Applied to Transaction Number and Receipt Currency Code.

Transaction Objects

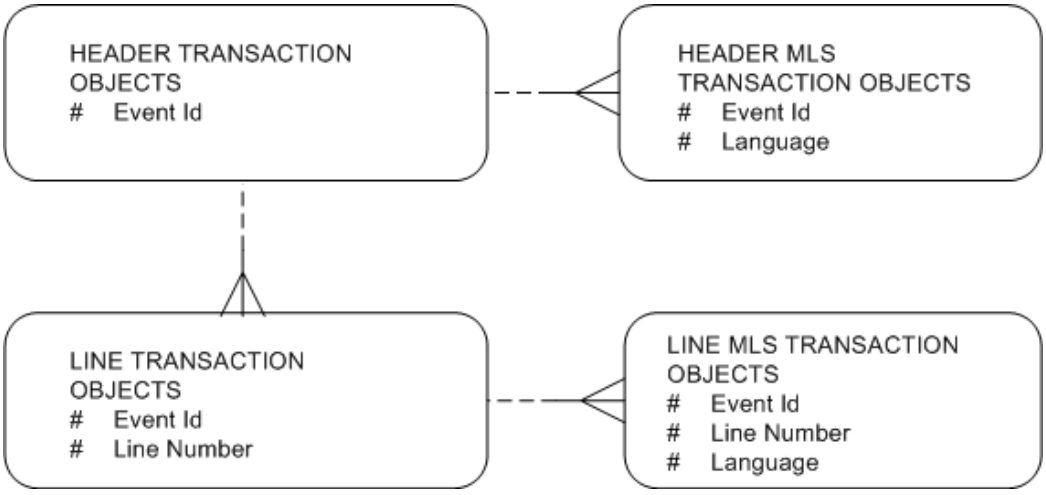
In addition to establishing what transaction attributes are available to the accounting program as sources, implementers determine which type of transaction objects object is appropriate to store them.

The Financial Services Accounting Hub supports the following types of transaction objects to store source values:

- Transaction Objects Headers
- Transaction Objects Multi Language Support Headers
- Transaction Objects Lines
- Transaction Objects Multi Language Support Lines

The name and number of transaction objects and their associated columns, with the exception of the primary keys, are determined by implementers. The relationships between different types of transaction objects and the primary key for each transaction object type are shown in the figure below and described in the Transaction Objects Types section, page 3-9.

Relationships Between Different Types of Transaction Objects



Transaction Objects and Event Classes

In the AMB, transaction objects are assigned to event classes. The list of source values available to the accounting program and therefore to users for inclusion in application accounting definitions can vary by event class. However, this does not mean that implementers need to create new transaction objects for each event class as a transaction object can be assigned to more than one event class.

Transaction Objects Types

This section describes the following transaction objects:

Header Transaction Objects

Implementers need to provide at least one header transaction object for each event class. Header transaction objects store one row with untranslated header source values for each accounting event. The primary key of a header transaction object is the event identifier.

Transaction details that are not translated and whose values do not vary by transaction line or distribution should normally be stored in header transaction objects. Examples of sources normally stored in header transaction objects include the Invoice Number for an invoice or the Bank Account Number for a receipt.

Header Multi-Language Support (MLS) Transaction Objects

Header MLS transaction objects are relevant to applications that support the Oracle E-Business Suite MLS feature. Header MLS transaction objects store translated header level source values. There should be one row per accounting event and language. The primary key of a header MLS transaction object is the event identifier and language.

Transaction details that are translated and whose values do not vary by transaction line or distribution are normally stored in header MLS transaction object columns. Examples include Payment Terms for a Payables invoice.

Implementers can avoid having to store source values in header MLS transaction objects by using value sets and lookup types.

See: Translated Sources, Lookup Types, and Value Sets Guideline, page 3-68

Line Transaction Objects

Line transaction objects are relevant to transaction types that can have more than a single line or distribution.

Line transaction objects store untranslated line level standard source values. There should be one row per distribution, identified by a unique line number. The primary key of a line transaction object is the event identifier and transaction object line number.

Transaction details that are not translated and whose values vary by transaction line or

distribution are normally stored in line transaction objects columns. Examples include the Invoice Number for a payment or the Invoice Line System Item for an invoice.

Line Multi-Language Support (MLS) Transaction Objects

Line MLS transaction objects are relevant to applications that support the Oracle Applications MLS feature. Line MLS transaction objects store translated line level standard source values. There should be one row per transaction distribution and language. The primary key of a Line MLS transaction objects is the event identifier, transaction object line number, and language.

Transaction details that are translated and whose values vary by transaction line or distribution should normally be stored in the transaction object in columns defined in a line MLS transaction object.

Transaction Object Columns

The transaction object columns must be one of the following:

- Primary Key Columns, page 3-10
- Standard Source Columns, page 3-11

Primary Key Columns

The appropriate primary key columns must be defined in all transaction objects. The table below lists the primary key column definitions.

Header Transaction Objects

Column Name	Description	Data Type	Comment
EVENT_ID	Event identifier	Integer	There should be one row per event.

Line Transaction Objects

Column Name	Description	Data Type	Comment
EVENT_ID	Event identifier	Integer	
LINE_NUMBER	Transaction objects line number	Integer	Unique within the event identifier (ID)

Header MLS Transaction Objects

Column Name	Description	Data Type	Comment
EVENT_ID	Event identifier	Integer	
LANGUAGE	Language code	Varchar(30)	Corresponds to the language column in _TL tables

Line MLS Transaction Objects

Column Name	Description	Data Type	Comment
EVENT_ID	Event identifier	Integer	
LINE_NUMBER	Transaction objects line number	Integer	
LANGUAGE	Language code	Varchar(30)	Corresponds to the language column in _TL tables

If a transaction object is implemented as a persistent or temporary table, the Financial Services Accounting Hub recommends creating a unique index that contains the primary key columns. For example, all line transaction objects should have a unique index on the event_id and line_number columns.

The accounting program uses the transaction objects primary keys to join transaction objects when selecting source values. As such, the line number primary key value should be consistent across line and MLS line transaction objects rows. Conversely, rows from both types of transaction objects with the same value for event_id and line_number should store source values that correspond to the same underlying transaction line or distribution.

Standard Source Columns

The majority of the columns in transaction objects are used to store standard sources. Transaction objects column names correspond to the AMB source codes.

Source codes and names should be unique across transaction and reference objects for an event class if they do not represent the same transaction attribute and there is the likelihood that they may have different values. For example, when transaction object 1 and transaction object 2 both have ENABLED_FLAG columns, the Create and Assign

Sources program creates a single source for ENABLED_FLAG. When the values of these columns for the two transaction objects are different, the extract may return incorrect values. In this case, the objects need to have unique column names.

The data type of the transaction object columns corresponds to source data types as shown in the table below.

Transaction Object Column and Source Data Type Correspondence

Transaction Object Column Data Type	Source Data Type
Number	Number
Date	Date
Varchar2	Alphanumeric
Char	Alphanumeric

The Financial Services Accounting Hub does not support the following data types in transaction objects and are ignored by the Create and Assign Sources program:

- Long
- Raw and Long Raw
- Rowid
- MLSLABEL

See: Create Sources and Assignments, page 3-14

Multiple Transaction Objects of the Same Type

For a single event class, the Financial Services Accounting Hub supports multiple objects of each type. There are several reasons why implementers may wish to define multiple transaction objects of the same type for a single event class:

- Sources common to several event classes

  A single transaction object can store sources that are common to several event classes. Such transaction objects can be assigned to multiple event classes. Sources that are specific to a single event class can be stored in a transaction object that is assigned only to that particular event class.

- Sources that are conditionally available

Different transaction objects store sources that are populated under different conditions. For example, a transaction object can store all sources that are populated when an optional foreign key is populated. Other transaction objects can store sources that are always available.

- Column limit

  If the number of sources that correspond for a transaction object type exceeds 1000, multiple transaction objects are necessary to store them.

Population of Transaction Objects

Not all transaction objects need to return rows for all transaction object headers and lines.

Always Populated Check Box

When a transaction object is assigned to an event class, define whether it is always populated.

See: To Set Up Accounting Event Class Options, page 5-2

The following additional rules apply to the Always Populated check box:

- For each event class, define at least one header transaction object as always populated.
- If line transaction objects are assigned to an event class, define at least one of them as always populated.
- If MLS header or MLS line transaction objects are assigned to an event class, it is not necessary for any of them to be always populated; however, there must be at least one corresponding header or line transaction object assigned to the same event class.

Transaction Objects Return No Rows

If a header or line level transaction object returns no rows for a transaction or transaction distribution respectively and it is marked as Always Populated, the accounting program returns an error.

Reference Objects

Use reference objects to define transaction objects without having SLA primary keys. To enable you to define relationships between reference objects, you can define up to two levels of reference objects, primary and secondary. The following steps describe how to define, maintain, and use reference objects.

1. In the Accounting Event Class Options window, define reference objects and define relationships between reference objects and transaction objects.

2. Run the Create and Assign sources program which completes the following tasks:

   - Creates sources for the application owning the reference object

   - Assigns source from a reference object to the event class to which the reference object is assigned

   Use reference objects to share sources across applications. For example, Payables implementers can assign sources from Oracle Projects to Payables event classes.

See: Create and Assign Sources Introduction, page 5-1

Create Sources and Assignments

Having created transaction objects and assigned them to event classes, run the Create and Assign Sources program. Based on the transaction objects definitions and corresponding database data dictionary definitions, the Create and Assign Sources program:

- Validates transaction objects

- Creates sources

- Assigns sources

- Duplicates column definitions

See: Create and Assign Sources Program, page 5-7

Transaction Objects Guidelines

For each eligible accounting event, source data from the transactions must be stored in transaction objects. The accounting program selects the source values it requires from these transaction objects to create subledger journal entries.

This section provides guidelines to help determine what transaction data should be stored in the transaction objects.

Introduction to Transaction Objects Guidelines

Source values are stored in transaction objects. The accounting program uses these source values and the application accounting definitions in the AMB to create subledger journal entries.

When determining what transaction object information should be defined as sources, consider including all relevant data that can potentially be used for creating application

accounting definitions. This includes data from your application as well as related applications. In general, all transaction data that users may find useful to create subledger journal entries should be made available as sources.

In analyzing the sources that should be made available to the accounting program, a good place to start is with the values referenced by the current programs used to create accounting. Values that determine how transactions are accounted prior to the uptake are used as sources that determine how the same types of transactions are accounted by the Financial Services Accounting Hub.

In many subledgers, transaction data is captured in a header and distribution lines format. As this data is the primary origin of information in the transaction, all of it should be made available as sources. While the header consists of information which is constant for the transaction, distribution data varies by line.

Summary Listing of Guidelines

When implementing the uptake of the Financial Services Accounting Hub consider the following guidelines when deciding on the sources to store in the transaction objects:

1. Accounting Attributes Guideline, page 3-17

   Accounting attributes values are needed by the accounting program to successfully create subledger journal entries.

2. Distribution Identifiers Guideline, page 3-48

   Distribution identifiers should be stored in the transaction object as accounting attributes.

3. Multiperiod Accounting Guideline, page 3-53

   To invoke multiperiod accounting for a transaction distribution, the transaction object should populate multiperiod accounting attributes with the appropriate values.

4. Monetary Amounts Guideline, page 3-54

   For any given accounting event, the sum of the monetary amounts stored in the transaction object should represent the total value of the underlying transaction in the subledger.

5. Gains and Losses Guideline, page 3-54

   The transaction object should not include explicit gain or loss transaction object lines. Instead, each transaction object line should include the ledger currency amounts and rate information for all related transactions for which gain or loss subledger journal lines are generated.

6. Foreign Currency Amounts Guideline, page 3-54

   Appropriate currency and rate information should be stored in the transaction

object.

7. Reporting Currency Amounts Guideline, page 3-59

   For each transaction object line, the reporting currency amounts and associated rate information for all primary and secondary ledgers, as well as for all reporting currencies, should be stored in the transaction objects.

8. Corrections for Rounding Guideline, page 3-60

   The accounting program detects rounding differences and accounts for them. Therefore, you do not need to include rounding distributions in the transaction objects.

9. Related Transactions, Lines, and Distributions Guideline, page 3-62

   The data from related transactions should be made available as sources. These transactions include applied to transactions, matched with transactions, and created from transactions. As an example, a Payables invoice line may require transactional information from the original purchase order.

10. Independence from Accounting Method Guideline, page 3-62

    The transaction object should be insensitive to the application accounting definitions that determine how transactions are accounted.

11. Balanced Debits and Credits Guideline, page 3-63

    Balanced debits and credits should not be included in the transaction object.

12. Granularity of Transaction Objects Lines Guideline, page 3-64

    For any transaction, the maximum level of detail should be stored in the transaction object and made available to the accounting program.

13. Third Party Control Accounts Guideline, page 3-66

    Third party data should be stored in the transaction object as accounting attributes.

14. Translated Sources, Lookup Types, and Value Sets Guideline, page 3-68

    The transaction object should store a translated source in each installed language used by ledgers linked to the accounting event's primary ledger.

15. Consistency Guideline, page 3-70

    Ensure that sources stored in the transaction object are stored in a consistent manner across different types of transactions.

16. Tax References Guideline, page 3-70

    The transaction object should store tax references as accounting attributes. The

Accounting Program uses tax references to establish a link between information stored in the tax repository and subledger journal entries.

17. Flexfields Guideline, page 3-71

    The transaction object should store flexfield column values as sources if they can be used in creating subledger journal entries.

18. Accounting Flexfield Derivation Mechanisms Guideline, page 3-72

    All values used to derive accounts prior to the uptake of the Financial Services Accounting Hub should be made available as sources.

19. Accounting Options Guideline, page 3-73

    Existing setup accounting options should be stored in the transaction object as sources.

20. Internal Identifiers Guideline, page 3-73

    In general, values that are never displayed to users should not be stored in the transaction object.

21. System Sources Guideline, page 3-74

    System sources are independently populated by the accounting program. They are not stored in the transaction object.

22. Accounting Reversals Guideline, page 3-78

    To invoke an accounting reversal for a transaction header or distribution, the transaction object should populate the appropriate header or line level accounting reversal options.

Accounting Attributes Guideline

The accounting program uses accounting attributes values to create subledger journal entries. The types of accounting attributes values are as follows:

- Values that are subject to special processing or values that are stored in named columns in journal entry headers and lines

  Examples of accounting attributes of this type are Entered Currency Code and Entered Amount.

- Values that control the behavior of the accounting program when processing a specific accounting event or transaction object line

  Examples of accounting attributes of this type are Accounting Reversal Indicator and Multiperiod Option.

Each accounting attribute is associated with a level:

- Header to create subledger journal entry headers
- Line to create subledger journal entry lines

The accounting attribute level should not be confused with the transaction object type. The accounting attribute level refers to the way in which the accounting program uses accounting attribute values to generate journal entries. The transaction object type refers to the kind of transaction object that is used to store the source values.

Accounting Attribute Assignments

The accounting program derives the values of accounting attributes by looking at the sources that are assigned to them. Almost all accounting attributes have sources assigned at the event class level. Depending on the accounting attribute, the accounting attribute assignment defaulted from the event class can be overridden on journal line types or application accounting definitions.

Event Classes

Assign sources to accounting attributes in the Accounting Attributes Assignments window. If the default assignment for an accounting attribute can be overridden on journal line types or application accounting definitions, assign one or more sources to the accounting attribute at the event class level. Designate one of the sources as the default.

If the source assigned to an accounting attribute cannot be overridden on journal line types or application accounting definitions, assign only one source to the accounting attribute at the event class level. In this case, the accounting attribute is not displayed in the Journal Line Types window or the Application Accounting Definitions window.

Journal Line Types

Assign sources to accounting attributes in the Journal Line Types window. The source assigned to an accounting attribute on a journal line type determines how the value of the accounting attribute is derived for all journal lines generated by the Financial Services Accounting Hub using that journal line type.

Journal line type accounting attributes are used in one of the following cases:

- When the accounting attribute value can be derived from more than one source

  For example, in a cross-currency receipt application transaction, the value of the accounting attribute Currency Code can vary from journal line to journal line within a single journal entry.

  The currency code accounting attribute can be derived from either the Invoice Currency Code or the Receipt Currency Code standard sources depending on the journal line type. Typically, the Receivable journal line type uses the Invoice Currency Code while the Cash Clearing journal line type uses the Receipt Currency Code.

- When the accounting attribute may be relevant to the processing of one journal line type and not to another

  For example, values for the accounting attribute Statistical Amount are relevant to journal line types used to record invoice expenses but not for journal line types used to record invoice liability.

  Implementers and users can define journal line type accounting attributes assignments.

Application Accounting Definitions

For each event class or event type, assign sources to accounting attributes in the Application Accounting Definitions window. The source assigned to an accounting attribute for an application accounting definition determines how the value of the accounting attribute is derived for all journal headers generated by the Financial Services Accounting Hub for events belonging to the event class or event type.

Use application accounting definition accounting attribute assignments when the accounting attribute value can be derived from one of many sources depending on accounting policies. For example, the accounting attribute GL Date for payment clear transactions can be derived from either the Event Date system source, the Clearance Date, or the Value Date standard sources.

Implementers and users can define application accounting definition accounting attribute assignments.

Accounting Attribute Levels and Transaction Object Types

The table below describes the relationships between transaction objects and accounting attribute levels.

Relationships Between Transaction Objects and Accounting Attribute Levels

Transaction Object Type	Accounting Attribute Level	Permitted	Comment
Header or MLS Header	Header	Yes	Sources defined in header or MLS header transaction objects can be used for journal entry header processing. For example, the standard source assigned to the accounting attribute Document Category Code can be defined in a header transaction object.
Header or MLS Header	Line	Yes	Sources defined in header or MLS header transaction objects can be used for journal entry line processing. For example, the standard sources assigned to the accounting attribute Entered Currency Code can be defined in a header transaction object.

Transaction Object Type	Accounting Attribute Level	Permitted	Comment
Line or MLS Line	Header	No	Sources defined in line or MLS line transaction objects cannot be used for journal entry header processing. For example, the standard source assigned to the accounting attribute Document Category Code cannot be defined in a line transaction object.
Line or MLS Line	Line	Yes	Sources defined in line or MLS line transaction objects can be used for journal entry line processing. For example, the standard sources assigned to the accounting attribute Entered Amount can be defined in a line transaction object.

Accounting Attributes

Accounting attributes are presented in functional groupings in table format. The table below describes the columns used in the tables.

Accounting Attributes Column Descriptions

Column Name	Description
Name	Displayed to implementers and users when creating accounting attribute assignments

Column Name	Description
Data Type	Only sources that have a compatible data type can be assigned to an accounting attribute.
Journal Entry Level	The accounting attribute level is header or line.
Accounting Method Builder Components	Accounting attributes can be assigned to sources for event classes, journal line types, or application accounting definitions.
Assignment Rules	Some accounting attributes must always be assigned to sources. Others must be assigned only if necessary to invoke a Financial Services Accounting Hub feature.
Optional or Mandatory	Assigning a source to an accounting attribute does not mean that the corresponding transaction object columns should necessarily have a value. Some accounting attributes are mandatory. Others are mandatory only if the corresponding subledger transaction values are populated.
Validation Rules	If populated, some sources assigned to accounting attributes are subject to special validation.

Entered Currency Accounting Attributes

The table below describes accounting attributes for entered currency.

Entered Currency Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Currency Code	Alphanumeric	Line	Event class and journal line type	Must be assigned	Mandatory	Must be a valid currency code

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Entered Amount	Number	Line	Event class and journal line type	Must be assigned	Mandatory	

Entered currency accounting attributes are relevant to all applications that use the Financial Services Accounting Hub. The accounting program uses them to populate the journal entry line entered currency code and amounts.

The entered currency accounting attributes must always be assigned to sources. The sources assigned to the entered currency accounting attributes must always contain a value. For event classes that support cross currency transactions and therefore, more than one entered currency and entered currency amount, more than one event class accounting attribute assignment is created.

Receivables Example: For the receipt application event class, the standard sources Invoice Currency Code and Receipt Currency Code are assigned to the accounting attribute Currency Code and one of them is designated the default accounting attribute assignment. Users choose which standard source to use depending on the journal line type.

See:

- Foreign Currency Amounts Guideline, page 3-54
- Reporting Currency Amounts Guideline, page 3-59

Ledger Currency Accounting Attributes

The table below describes accounting attributes for ledger currency.

Ledger Currency Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Conversion Rate Type	Alphanumeric	Line	Event class and journal line type	Should be assigned	Optional	Should be a valid General Ledger conversion rate type or User
Conversion Date	Date	Line	Event class and journal line type	Should be assigned	Mandatory if the conversion rate type is not User	
Conversion Rate	Number	Line	Event class and journal line type	Should be assigned	Mandatory if the conversion rate type is User	
Accounted Amount	Number	Line	Event class and journal line type	Must be assigned if the Financial Services Accounting Hub does not calculate accounted amounts	Mandatory if the Financial Services Accounting Hub does not calculate accounted amounts	

Ledger currency accounting attributes are relevant to all applications that use the accounting program. The accounting program uses them to populate journal entry accounted amounts.

If a transaction's entered currency is different from the ledger currency, the accounting program copies the conversion date, conversion rate, and conversion rate type to the corresponding journal entry lines. If the entered currency is the same as the ledger currency, the accounting program ignores the conversion type and conversion rate.

For event classes that support foreign currency transactions and therefore more than one exchange rate and reporting currency amount, more than one event class accounting attribute assignment is created.

Receivables Example: For the receipts and receipt application event class, the standard

sources Invoice Accounted Amount and Receipt Accounted Amount are assigned to the accounting attribute Accounted Amount and one of them is designated the default accounting attribute assignment. Users choose which standard source to use depending on the journal line type.

Payables Example: For a payment clear event class, the standard sources Invoice Exchange Rate, Payment Exchange Rate, and Clear Exchange Rate are assigned to the accounting attribute Conversion Rate and one of them is designated the default accounting attribute assignment. Users choose which standard source to use depending on the journal line type.

See:

- Foreign Currency Amounts Guideline, page 3-54
- Reporting Currency Amounts Guideline, page 3-59

Gain/Loss Accounting Attributes

The table below describes gain/loss accounting attributes.

Gain/Loss Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Exchange Gain Account	Number	Header	Event class	Must be assigned if there is exchange gain on the transaction but no gain/loss journal line type is assigned to the journal line definition.	Optional	

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Exchange Loss Account	Number	Header	Event class	Must be assigned if there is exchange loss on the transaction but no gain/loss journal line type is assigned to the journal line definition.	Optional	
Gain or Loss Reference		Line	Event class	Must have same value for each set of debit and credit lines.	Required only if debit and credit side entries are passed to the Financial Services Accounting Hub as separate transaction object lines and there is exchange /gain loss on the transaction.	

When the Financial Services Accounting Hub calculates gain/loss amount, if no gain/loss journal line type is defined then the Exchange Gain/Loss accounts are used.

The Gain/Loss Reference accounting attribute groups entry lines together when

calculating gain/loss. The Financial Services Accounting Hub combines the accounted debit and accounted credit amounts for lines with the same gain/loss reference and compares the total of accounted debit and total of accounted credit to calculate the exchange gain/loss.

Distribution Identifier Accounting Attributes

The table below describes distribution identifier accounting attributes.

Distribution Identifier Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Distribution Type	Alphanumeric	Line	Event class	Must be assigned	Mandatory	
First Distribution Identifier	Alphanumeric	Line	Event class	Must be assigned	Mandatory	
Second Distribution Identifier	Alphanumeric	Line	Event class	Optional	Optional	
Third Distribution Identifier	Alphanumeric	Line	Event class	Should not be assigned if second distribution identifier is not assigned	Optional	
Fourth Distribution Identifier	Alphanumeric	Line	Event class	Should not be assigned if third distribution identifier is not assigned	Optional	
Fifth Distribution Identifier	Alphanumeric	Line	Event class	Should not be assigned if fourth distribution identifier is not assigned	Optional	

Distribution identifiers accounting attributes are relevant to all applications. The distribution identifier information links subledger transaction distributions to their corresponding journal entry lines. In addition, many of the Financial Services Accounting Hub features, including accounting reversals, rely on the correct definition and storing of distribution identifiers in the line transaction objects.

The distribution type and first distribution identifiers are always assigned to sources. If a transaction distribution is identified by a composite primary key, additional distribution identifiers are assigned to standard sources, as appropriate. Values for the distribution type and distribution identifiers are always stored in accounting transaction objects. The combination of the values of the system transaction identifiers with the values of the distribution identifiers uniquely identify a subledger transaction distribution line.

See: Distribution Identifiers Guideline, page 3-48

Line Accounting Reversal Accounting Attributes

The table below describes line accounting reversal accounting attributes.

Line Accounting Reversal Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Accounting Reversal Indicator	Alphanumeric	Line	Event class	Should be assigned if another line accounting reversal accounting attribute is assigned	Optional	Should be: Y – reversal transaction object line B – reversal and standard transaction object line N or null – standard transaction object line

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Accounting Reversal Distribution Type	Alphanumeric	Line	Event class	Should be assigned if another line accounting reversal accounting attribute is assigned	Mandatory if the accounting reversal indicator is populated	
Accounting Reversal First Distribution Identifier	Alphanumeric	Line	Event class	Should be assigned if another line accounting reversal accounting attribute is assigned	Mandatory if the accounting reversal indicator is populated	
Accounting Reversal Second Distribution Identifier	Alphanumeric	Line	Event class	Should not be assigned if reversal first distribution identifier is not assigned	Mandatory if the corresponding second distribution identifier is populated	
Accounting Reversal Third Distribution Identifier	Alphanumeric	Line	Event class	Should not be assigned if reversal second distribution identifier is not assigned	Mandatory if the corresponding third distribution identifier is populated	
Accounting Reversal Fourth Distribution Identifier	Alphanumeric	Line	Event class	Should not be assigned if reversal third distribution identifier is not assigned	Mandatory if the corresponding fourth distribution identifier is populated	

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Accounting Reversal Fifth Distribution Identifier	Alphanumeric	Line	Event class	Should not be assigned if reversal fourth distribution identifier is not assigned	Mandatory if the corresponding fifth distribution identifier is populated	

Line accounting reversal accounting attributes are relevant to applications that wish to take advantage of the accounting reversal feature. The accounting program uses them to identify transaction (distributions) whose accounting impact should be reversed.

For the accounting program to successfully create a line accounting reversal, the accounting reversal indicator, distribution type, and first distribution identifier should always be assigned to sources.

The definition of the accounting reversal distribution type and distribution identifiers mirrors the definition of the distribution identifiers.

See: Accounting Reversals Guideline, page 3-78

Transaction Accounting Reversal Accounting Attributes

The table below describes transaction accounting reversal accounting attributes.

Transaction Accounting Reversal Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Transaction Accounting Reversal Indicator	Alphanumeric	Header	Event class	Optional	Optional	Should be: Y – reversal transaction object header N or null – standard transaction object header

The transaction accounting reversal accounting attribute is relevant to applications that wish to take advantage of transaction level accounting reversals feature. The accounting program uses them to invoke the reversal of all accounting generated for the transaction associated with the accounting event.

For the accounting program to successfully create a transaction accounting reversal, the accounting reversal indicator should be assigned to a source.

See: Accounting Reversals Guideline, page 3-78

GL Date Accounting Attributes

The table below describes GL date accounting attributes.

GL Date Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
GL Date	Date	Header	Event class and application accounting definition	Must be assigned	Mandatory	Should be in open GL period
Accrual Reversal GL Date	Date	Header	Event class and application accounting definition	Optional	Optional	Should be later than the GL date

The GL Date accounting attribute is relevant to all applications. The accounting program uses it to derive the GL date of journal entries. Normally, the event date system source is the only source assigned to the GL Date accounting attribute.

The Accrual Reversal GL Date accounting attribute is relevant to applications using the accrual reversal feature.

Users can assign system and any standard date source to the Accrual Reversal GL Date in the Accounting Attribute Assignments window. The system sources are as follows:

- Next Day

  The GL date of the accrual reversal will be the day following the GL date of the accrual entry. If there is a transaction calendar assigned to the ledger, this will be the next business day.

- First Day Next GL Period

The GL date of the accrual reversal entry will be the first day of the following period. If there is a transaction calendar assigned to the ledger, this will be the first business day.

- Last Day Next GL Period

  The GL date of the accrual reversal entry will be the last day of the following period. If there is a transaction calendar assigned to the ledger, this will be the last business day.

When the accrual reversal GL date accounting attribute returns a value, the Create Accounting program generates an entry that reverses the accrual entry.

See: Accounting Reversals Guideline, page 3-78

Third Party Accounting Attributes

The table below describes third party accounting attributes.

Third Party Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Party Type	Alphanumeric	Line	Event class	Should be assigned if another third party accounting attribute is assigned	Mandatory if the party identifier is populated	Should be C for customer or S for supplier
Party Identifier	Number	Line	Event class and journal line type	Should be assigned if another third party accounting attribute is assigned	Mandatory if the party type is populated	If party type is C, should be a valid customer account; if party type is S, should be a valid supplier identifier

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Party Site Identifier	Number	Line	Event class and journal line type	Should be assigned if another third party accounting attribute is assigned	Should not be populated unless the party type and party identifier are populated.	If party type is C, should be a valid customer account; if party type is S, should be a valid supplier site

Third party accounting attributes are relevant to subledger applications that use third party control accounts. The third party accounting attributes link suppliers and customers to their corresponding subledger journal entry lines in the supplier and customer subledgers. For all subledger transactions that represent financial transactions with third parties, all third party accounting attributes have sources assigned. If a transaction line is associated with a customer or supplier, the transaction objects need to include values for all sources mapped to third party accounting attributes for the event class.

Some types of transactions involve more than one third party. For these types of transactions, more than one event class accounting attribute assignment is necessary to support the accounting for third party control accounts.

Receivables Example: Since a receipt and the invoices that it is applied to can have different customers or customer sites, the Bill To Customer Account Identifier and Pay to Customer Account Identifier are both defined as standard sources and assigned to the third party accounting attribute Party Identifier in the Journal Line Accounting Attribute Assignments window. One of them is designated the default accounting attribute assignment. Users choose which standard source to use depending on the journal line type.

See:

- Third Party Control Accounts Guideline, page 3-66
- Defining Segment Qualifiers, *Oracle General Ledger User Guide*

Statistical Amount Accounting Attribute

The table below describes the statistical amount accounting attribute.

Statistical Amount Accounting Attribute

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Statistical Amount	Number	Line	Event class and journal line type	Optional	Optional	

The statistical amount is relevant to applications that currently pass statistical amounts to General Ledger. It is stored on subledger journal entry lines and subsequently passed to General Ledger for statistical analysis.

Tax References Accounting Attributes

The table below describes tax references accounting attributes.

Tax References Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Detail Tax Line Reference	Number	Line	Event class and journal line type	Optional	Optional	
Detail Tax Distribution Reference	Number	Line	Event class and journal line type	Optional	Optional	
Summary Tax Line Reference	Number	Line	Event class and journal line type	Optional	Optional	

The tax reference accounting attributes are relevant to applications that uptake the tax initiative. The tax team uses the tax reference accounting attributes to link subledger transaction tax distributions to their corresponding journal entry lines.

The Oracle E-Business Tax development team specifies which tax reference values are mandatory in transaction objects and should be assigned to standard sources.

See: Tax References Guideline, page 3-70

Reconciliation Reference Accounting Attribute

The table below describes the reconciliation reference accounting attribute.

Reconciliation Reference Accounting Attribute

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Reconciliation Reference	Alphanumeric	Line	Journal line type	Optional	Optional	

The reconciliation reference is relevant to all applications. However, since users have full control over which sources are used as the reconciliation reference, it is not necessary to assign a source to the accounting attribute at the event class level. The reconciliation reference is displayed only on journal line types. When users create journal line types, they can assign any of the sources assigned to the event class to the reconciliation reference.

The reconciliation reference is stored on subledger journal entry lines and subsequently passed to General Ledger for use in the journal entry reconciliation feature.

Budgetary Control Accounting Attributes

The table below describes budgetary control accounting attributes.

Budgetary Control Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Budget Version Identifier	Number	Header	Event class	Should be assigned for event classes that support budgets	Mandatory for events with budgetary amounts	Should be a valid General Ledger budget identifier

The budget version identifier is relevant to applications that support budgetary journals.

To assign the budget version identifier to a standard source and create budget journal

line types, the event class must be enabled for budget entries. For the accounting program to successfully create budget journals, there must be a valid budget version identifier.

See: Allocation Accounting Attributes, page 3-45

Document Sequencing Accounting Attributes

The table below describes document sequence accounting attributes.

Document Sequence Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Document Sequence Category Code	Alphanumeric	Header	Event Class	Should be assigned if another document sequence accounting attribute is assigned	Mandatory if document sequence value is populated	Should be a valid sequence category
Document Sequence Identifier	Number	Header	Event class	Should be assigned if another document sequence accounting attribute is assigned	Mandatory if document sequence value is populated	Should be a valid document sequence identifier
Document Sequence Value	Number	Header	Event class	Should be assigned if another document sequence accounting attribute is assigned	Mandatory if document sequence version identifier is populated	

The document sequence accounting attributes are relevant to applications that use the Application Object Library document sequencing feature to assign sequence numbers to subledger transactions. The accounting program uses them to provide a user link between subledger transactions and their corresponding subledger journal entries.

Assign all document sequence accounting attributes to sources or do not assign any. In addition, the Document Sequence Category Code is made available as an Accounting Sequence Numbering control attribute.

Either none of the document sequence accounting attributes should be assigned to standard sources or all document sequence accounting attributes should be assigned to standard sources.

Multiperiod Accounting Attributes

The table below describes multiperiod accounting attributes.

Multiperiod Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Multiperiod Option	Alphanumeric	Line	Event class	Should be assigned if another multiperiod accounting attribute is assigned	Optional	Should be Y for yes or N for no
Multiperiod Start Date	Date	Line	Event class	Should be assigned if another multiperiod accounting attribute is assigned	Mandatory if multiperiod option is populated	
Multiperiod End Date	Date	Line	Event class	Should be assigned if another multiperiod accounting attribute is assigned	Mandatory if multiperiod period type is not populated	

The multiperiod accounting attributes are relevant to applications providing the multiperiod accounting feature to their customers.

Either no multiperiod accounting attributes should be assigned to sources or all multiperiod accounting attributes should be assigned to sources.

See: Multiperiod Accounting Guideline, page 3-53

Transfer to GL Accounting Attribute

The table below describes the Transfer to GL accounting attribute.

Transfer to GL Accounting Attribute

Name	Data Type	Journal Entry Level	Accounting Method Builder Components	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Transfer to GL Indicator	Alphanumeric	Header	Event class	Optional	Optional	Should be Y or N

The transfer to GL accounting attribute is relevant only to applications that wish to generate historical journal entries using the Financial Services Accounting Hub. The Transfer to GL process uses this accounting attribute to determine whether to transfer subledger journal entries to General Ledger.

If the transfer to GL accounting attribute is not assigned to a source, the Transfer to GL process transfers journal entries for the event class to General Ledger.

If the transfer to GL accounting attribute is assigned to a source and the source is not populated, the Transfer to GL process transfers journal entries for the event class to General Ledger.

Transaction Rounding Reference Accounting Attribute

The transaction rounding reference determines the level at which the transaction rounding difference is calculated. It is used along with the rounding class to detect and correct subledger transaction rounding differences. If a transaction rounding reference is not provided, transaction rounding is not calculated.

Subledger transaction rounding differences occur when the sum of the accounted amounts for a subledger transaction is not equal to the sum of the entered amounts multiplied by the conversion rate. Rounding differences can also occur on functional currency transactions when the sum of an unrounded entered amount after rounding the minimum accountable units and precision of the currency does not equal the final rounded entered amount. The accounting program compares the sum of the rounded accounted amounts with the sum of the unrounded accounted amounts by transaction rounding reference and rounding class to detect subledger transaction rounding differences.

Transaction rounding differences are corrected by adding the rounding difference amount to one of the subledger journal entry lines with the same transaction rounding

difference and rounding class. Specifically, it is added to the journal entry line with the largest amount.

Subledger Transaction Rounding Example

The subledger transaction rounding example describes a journal entry with a subledger transaction rounding difference and illustrates how the rounding class is used to detect and correct the rounding difference.

For this example, the table below describes the accounting attributes and the sources that are mapped for the journal line types Invoice Liability and Expense.

Accounting Attribute	Source
Entered Amount	Entered Amount
Conversion Date	GL Date
Conversion Rate Type	Invoice Conversion Rate Type
Transaction Rounding Reference	Invoice Identifier

The rounding class for the journal line type Invoice Liability is Liability. The rounding class for the journal line type Expense is Expense.

The table below describes a Payables invoice with entered currency SEK and a conversion rate of 0.1. The invoice identifier (transaction rounding reference) is 40067.

Invoice 123 from Supplier ABC, 18-JAN-2003

Number	Description		Invoice Entered Amount
1	item		10.04
2	item		10.04
		Invoice Total (SEK):	20.08

The table below describes the temporary journal entry records for the Payables invoice. The following information applies to this example:

- The entered currency is SEK.
- The conversion rate is 0.1.

- The transaction rounding reference is 400067.

Line Number	Account	Rounding Class	Entered DR	Entered CR	Accounted DR (USD)	Accounted CR (USD)
1	01-Expense	Expense	10.04		1.00	
2	02-Expense	Expense	10.04		1.00	
3	01-Liability	Liability		10.04		1.00
4	02-Liability	Liability		10.04		1.00

Since lines 1 and 2 have the same rounding class and transaction rounding reference, they are grouped together for comparison for subledger transaction rounding.

Although the journal entry is balanced, a rounding difference has occurred since the sum of the entered amounts multiplied by the conversion rate is $2.01, but the sum of the accounted amounts for the lines is only $2.00.

The accounted amount for the subledger transaction is calculated as described in the table below.

Subledger Transaction Entered Amount	X	Conversion Rate	=	Unrounded Accounted Amount	Rounded Accounted Amount
20.08	X	0.10	-	2.008	2.01

The accounted amount for the individual lines is calculated as described in the table below.

Subledger Transaction Entered Amount	X	Conversion Rate	=	Unrounded Accounted Amount	Rounded Accounted Amount
10.04	X	0.10	=	1.004	1.00
10.04	X	0.10	=	1.004	1.00
					Total: 2.00

Since lines 3 and 4 have the same rounding class and transaction rounding reference, a similar calculation is performed as described above and a rounding difference of 0.01 is calculated.

The Financial Services Accounting Hub corrects the transaction rounding differences and the table below describes the final journal entry.

Account	Rounding Class	Entered DR	Entered CR	Accounted DR (USD)	Accounted CR (USD)
01-Expense	Expense	10.04		1.00	
02-Expense	Expense	10.04		1.01	
01-Liability	Liability		10.04		1.00
02-Liability	Liability		10.04		1.01

A rounding amount of $0.01 is added to the 02-Expense. A rounding amount of $0.01 is added to the 02-Liability account.

Rounding Class Example

Using the same Payables invoice from the Subledger Transaction Rounding example, assume that the rounding class is Expense for both journal line types Invoice Expense and Invoice Liability.

Although the journal entry has the same amounts as the previous example, the rounding class is the same for all four lines. Since the sum of the accounted amount for the lines with the same rounding class and transaction reference is zero, and the sum of entered amount from the lines with the same rounding class and transaction rounding reference multiplied by conversion rate is also zero, no rounding reference is detected.

The table below describes the final journal entries for the invoice. The following information applies to the example:

- The rounding class is Liability.
- The entered currency is SEK.
- The conversion rate is 0.1.
- The transaction rounding reference is 400010.

Line Number	Account	Rounding Class	Entered DR	Entered CR	Accounted DR (USD)	Accounted CR (USD)
1	01-Expense	Expense	10.04		1.00	
2	02-Expense	Expense	10.04		1.00	
3	01-Liability	Expense		10.04		1.00
4	02-Liability	Expense		10.04		1.00

Business Flows Accounting Attributes

The table below describes business flows accounting attributes.

Business Flows Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Applied To Amount	Number	Line	Event class and journal line type		Mandatory if the currency for prior entry is different than the currency currently selected	Validation only takes place if validation an amount is previously enabled.
Applied to Application ID	Number	Line	Event class and journal line type	Optional	Optional	Must be a valid application ID
Applied to Entity Code	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	Must be a valid Entity for the application selected in Applied to Application ID

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Applied to First System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	
Applied to Second System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to First System Transaction Identifier is not assigned	Optional	
Applied to Third System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Second System Transaction Identifier is not assigned	Optional	
Applied to Fourth System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Third System Transaction Identifier is not assigned	Optional	
Applied to Distribution Type	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	
Applied to First Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Applied to Second Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to First Distribution Identifier is not assigned	Optional	
Applied to Third Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Second Distribution Identifier is not assigned	Optional	
Applied to Fourth Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Third Distribution Identifier is not assigned	Optional	
Applied to Fifth Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Fourth Distribution Identifier is not assigned	Optional	

The business flow accounting attributes are referred to as applied to accounting attributes. If a transaction is applied to a prior transaction in the business flow, the transaction object must populate sources assigned to applied to accounting attributes with sufficient information to allow the accounting program to uniquely identify a transaction object line for a prior event in the business flow. When deriving accounting data from a previous event in the business flow, the accounting program searches for a journal entry line for the prior event using a combination of the applied to accounting attributes and the business flow class of both journal entries.

The Applied To Amount accounting attribute is used to calculate the accounted amount and gain/loss in cross-currency applications when business flows are implemented. This

attribute value is used to calculate the accounted amount when a source is mapped to the Applied To Amount attribute on a journal line type and the entered currency is different than the original currency entered.

> **Note:** When using business flows, certain accounting attribute values are unavailable for source assignment in the Accounting Attributes Assignment window because they will be copied from the prior journal entry.

Allocation Accounting Attributes

The table below describes allocation accounting attributes.

Allocation Accounting Attributes

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Allocated to Application ID	Number	Line	Event class and journal line type	Optional	Optional	Must be a valid application ID
Allocated to Entity Code	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	Must be a valid Entity for the application selected in Applied to Application ID
Allocated to First System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	
Allocated to Second System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to First System Transaction Identifier is not assigned	Optional	

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Allocated to Third System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Second System Transaction Identifier is not assigned	Optional	
Allocated to Fourth System Transaction Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Third System Transaction Identifier is not assigned	Optional	
Allocated to Distribution Type	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	
Allocated to First Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must be assigned if Applied to Application ID is assigned	Mandatory if Applied to Application ID is populated	
Allocated to Second Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to First Distribution Identifier is not assigned	Optional	

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Allocated to Third Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Second Distribution Identifier is not assigned	Optional	
Allocated to Fourth Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Third Distribution Identifier is not assigned	Optional	
Allocated to Fifth Distribution Identifier	Alphanumeric	Line	Event class and journal line type	Must not be assigned if Applied to Fourth Distribution Identifier is not assigned	Optional	
Actual Upgrade Option	Alphanumeric	Line	Event Class	Should be assigned to use Actual Upgrade accounting attributes to derive a reversal accounting entry or a business flow accounting line	Optional	

Name	Data Type	Journal Entry Level	Accounting Method Builder Component	Accounting Attribute Assignment Rules	Optional or Mandatory	Validation Rules
Encumbrance Upgrade Option	Alphanumeric	Line	Event Class	Should be assigned to use Encumbrance Upgrade accounting attributes to derive a reversal accounting entry or a business flow accounting line	Optional	

The allocation accounting attributes are always enabled and they are used in tracking transactions that are allocated to other transactions. No accounting is derived from these attributes. These attributes are also used for the following:

- Grouping transactions by the budgetary control code
- Enabling both allocations and business flows to coexist in the same transaction line

  For example, burdening that is computed on the raw cost coming from an invoice is allocated to the invoice, yet both the invoice and the burdening may have a business flow relative to a purchase order or a requisition.

Distribution Identifiers Guideline

Distribution identifiers are stored in the transaction object and must be mapped to accounting attributes.

Distribution identifiers are the internal (surrogate) keys that uniquely identify the transaction distribution associated with a transaction object line.

The purpose of distribution identifiers is:

- To establish a link between transaction distributions and subledger journal entry lines by creating reconciliation reports between the subledger transactions and their corresponding accounting

  See: Distribution Link Example, page 3-50

- To support custom sources

Since the distribution identifier represents the lowest level of detail available in the transaction data model, it is possible to create PL/SQL functions that return any data item related to the transaction distribution, transaction, or set up.

See: Custom Sources, *Oracle Subledger Accounting Implementation Guide*

- To support the accounting reversals feature by identifying distributions whose accounting should be reversed

  See: Accounting Reversals Guideline, page 3-78

- To support the business flows feature by identifying journal entries for prior transactions in the business flow

  See: Business Flows, *Oracle Subledger Accounting Implementation Guide*

Distribution Link Type

All distribution identifiers have a distribution link type to differentiate between distributions. This is a mandatory source and must always be populated by the Load Transaction Objects program.

The distribution link type ensures that other distribution identifiers uniquely identify the underlying transaction's distribution associated with a transaction object line. Implementers should use a distribution link type source analogous to a table alias.

Each of the distribution identifiers is populated for the corresponding distribution link type.

Receivables Example: The distribution link type distinguishes between transaction distributions stored in the transactions distribution data model and those stored in the receipt or adjustments distribution data model.

Payables Example: The distribution link type distinguishes between transaction distributions stored in the invoice distribution data model and those stored in the payment distribution data model.

Distribution Identifiers

The Financial Services Accounting Hub supports up to five distribution identifiers for each distribution link type. The distribution link type is the same for all transaction object lines for an accounting event and therefore should normally be mapped to a header level source.

Payables Example: The table below lists the components for distribution identifiers that can be defined for the invoice event entity.

Source Name	Source Value	Accounting Attribute Name
Distribution Type	AP_INV	Distribution Link Type
Invoice Distribution Detail Identifier	12345	First Distribution Identifier

Receivables Example: The table below lists the components for distribution identifiers that can be defined for the transaction event entity.

Source Name	Source Value	Accounting Attribute Name
Transaction Type	AR_TRANS	Distribution Link Type
Transaction Line Identifier	23456	First Distribution Identifier
Transaction Line Detail Identifier	2	Second Distribution Identifier

Distribution Link Example

When the accounting program creates a subledger journal entry line, it inserts a row into the distribution link table for each transaction object line that contributes to it.

The example below shows how the distribution link table is populated. Assume that the event entity Transactions has the First Distribution Identifier accounting attribute mapped to the Transaction Distribution Identifier standard source.

The example is illustrated in the tables below:

- Transaction Header table and Distribution table (owned by subledger applications)
- Transaction Objects Lines table (owned by subledger applications)
- Subledger Journal Entry Lines table (owned by the Financial Services Accounting Hub)
- Financial Services Accounting Hub Distribution Link Table Lines table (owned by Financial Services Accounting Hub)

The table below shows the header details for a Receivables transaction.

Transaction Header

Customer Name / Number	Transaction Type	Transaction Number	GL Date	VAT Date
ABC Ltd./123	Domestic Invoice	A-1234	02-Jan-2002	30-Dec-2001

The table below shows the distribution details for the same transaction.

Transaction Distribution

Transaction Distribution Identifier	Transaction Line Number	Distribution Detail Number	Transaction Line Type	Amount	Description
1234	1	1	Item	1000	Standard Item
1235	1	2	Tax	160	VAT @16%
1236	2	1	Item	2000	Standard Item
1237	2	2	Tax	500	VAT @25%
1238	3	1	Item	3000	Standard Item
1239	3	2	Tax	480	VAT @16%

The table below shows the details of the transaction object lines for the transaction.

Transaction Object Lines

Event ID	Transaction Objects Line No.	Transaction Distribution Identifier	Transaction Line Number	Distribution Detail Number	Distribution Type	Amount	Description
1	1	1234	1	1	Item	1000	Standard Item

Event ID	Transaction Objects Line No.	Transaction Distribution Identifier	Transaction Line Number	Distribution Detail Number	Distribution Type	Amount	Description
2	2	1235	1	2	Tax	160	VAT @16%
3	3	1236	2	1	Item	2000	Standard Item
4	4	1237	2	2	Tax	500	VAT @25%
5	5	1238	3	1	Item	3000	Standard Item
6	6	1239	3	2	Tax	480	VAT @16%

The table below shows the subledger journal entry lines for the above invoice.

Subledger Journal Entry Lines

Entry Header Identifier	Entry Line Number	Ledger (ID)	Accounting Class	Account	Debit	Credit	Description
6000	1	101	Receivable	01.4300.000	7140		All lines
6000	2	101	Revenue	01.6000.000		4000	Transaction Objects lines 1 + 5
6000	3	101	Revenue	01.6001.000		2000	Transaction Objects line 3
6000	4	101	Tax	01.4372.000		1140	Transaction Objects lines 2 + 4 + 6

The table below records how each of the transaction object amounts have contributed to the subledger journal entry lines.

Financial Services Accounting Hub Distribution Link Table Lines

Entry Header Identifier	Journal Entry Line No.	Ledger (ID)	First Distribution Identifier	Entered Amount	Ledger Amount
6000	1	101	1234	1000	1000
6000	1	101	1235	160	160
6000	1	101	1236	2000	2000
6000	1	101	1237	500	500
6000	1	101	1238	3000	3000
6000	1	101	1239	480	480
6000	2	101	1234	1000	1000
6000	2	101	1238	3000	3000
6000	3	101	1236	2000	2000
6000	4	101	1235	160	160
6000	4	101	1237	500	500
6000	4	101	1239	480	480

Multiperiod Accounting Guideline

Multiperiod accounting provides accounting for transaction distributions to be spread over one or more accounting periods.

The features of multiperiod accounting depend on a combination of accounting attribute values and application accounting definitions created in the AMB. To invoke multiperiod accounting, the transaction object should store source values relating to multiperiod accounting and these sources should be mapped to the multiperiod accounting attributes.

Monetary Amounts Guideline

The accounting program uses the entered and ledger currency amounts to create subledger journal entry debit and credit amounts. The following guidelines apply to monetary amounts and are described in this section:

- Currency Precision, page 3-54
- Monetary Amount Sign (positive or negative), page 3-54

Currency Precision

It is not necessary to round entered amount or ledger currency amount in the transaction objects. The Financial Services Accounting Hub rounds the amount in the final journal entries based on the precision and minimum accountable unit of the currency.

Monetary Amount Sign (positive or negative)

Users can view underlying transactions in transaction windows. In general, the sign of a monetary amount stored in the transaction object for an accounting event should be the same as that which is displayed to users. For example, if both invoices and receipts are displayed to users as positive amounts in transaction windows and on reports, then positive amounts should also be stored in the transaction object.

It increases understanding if amounts that represent reversals or cancellations of earlier transactions are represented with the opposite sign of the original amount. For example, an invoice cancellation amount should be negative if the corresponding invoice amount is positive.

Gains and Losses Guideline

If Calculate Gain or Loss Amounts is enabled for the event class, the transaction object should not include explicit gain or loss lines. The gain or loss amount is calculated by the Financial Services Accounting Hub by subtracting the total unrounded credit amounts from the total unrounded debit amounts. If it results in a positive amount, a loss line is created; otherwise, a gain line is created according to the defined journal line type.

If Calculate Gain or Loss Amounts is disabled for the event class, an explicit gain or loss transaction object line is needed if a gain or loss journal line needs to be generated.

Foreign Currency Amounts Guideline

Appropriate currency and rate information should be stored in the transaction object. The following accounting attributes relate to foreign currency transactions:

- Entered Currency Code
- Entered Amount
- Accounted Amount
- Conversion Rate
- Conversion Date
- Conversion Rate Type

The following sections describe two distinct cases of foreign currency transactions. They contain examples of how the accounting attribute assignments can be used to create the appropriate subledger journal entries in each case.

The ledger currency is made available to the accounting program as a system source.

See:

- Standard, System, and Custom Sources, page 3-6
- System Sources Guideline, page 3-74

If the Financial Services Accounting Hub is not configured to calculate accounted amount, the accounted amount accounting attribute must be mapped to a source. If the Financial Services Accounting Hub is configured to calculate accounted amounts, it does not use the accounted amount provided in the transaction object and it does not generate warnings even if there is no value in the source mapped to this accounting attribute.

See: Override Calculated Accounted Amount, page 15-2

From the perspective of the Financial Services Accounting Hub, the two distinct cases of foreign currency transactions are:

- Stand alone foreign currency transactions: foreign currency transactions that are not applied to other transactions, page 3-55
- Foreign currency transactions that are applied to other transactions, page 3-57

  These transactions give rise to the following scenarios:

  - Foreign currency transactions applied to transactions of the same currency
  - Cross-currency transactions: transactions applied to transactions of a different currency

Foreign Currency Transactions Not Applied to Other Transactions

In cases where foreign currency transactions are not applied to other transactions, no

gain or loss is created and all journal line types can use the same accounting attribute.

Receivables Example: Assuming that Calculate Accounted Amount is disabled, an invoice of GBP 110 is recorded in the system on January 18th with a rate of 1.60 (USD 176).

The table below lists source values that should be stored in the transaction object.

Invoice Transaction Object Lines

Line No.	Line Type	Entered Currency Code	Entered Amount	Ledger Currency Amount	Conversion Rate	Conversion Date	Conversion Rate Type
1	Item	GBP	100.00	160.00	1.60	18-Jan-05	User
2	Tax	GBP	10.00	16.00	1.60	18-Jan-05	User

The table below lists accounting attribute assignments for all journal line types for the Invoice event class.

Accounting Attribute Assignments for all Journal Line Types for the Invoice Event Class

Accounting Attribute Name	Standard Source
Entered Currency Code	Entered Currency Code
Entered Amount	Entered Amount
Accounted Amount	Ledger Currency Amount
Conversion Rate	Conversion Rate
Conversion Date	Conversion Date
Conversion Rate Type	Conversion Rate Type

Based on the standard source values, the accounting attribute assignment and some simple application accounting definitions, the table below lists details of the resulting subledger journal entry. The conversion rate is 1.60.

Invoice Issued Subledger Journal Entry

Entry Line No.	Journal Line Type	Account	Entered Currency Code	Entered Amount	Accounted DR (USD)	Accounted CR (USD)
1	Receivable	01.6000.001	GBP	110.00	176.00	
2	Revenue	01.4300.000	GBP	100.00		160.00
3	Tax	01.4700.000	GBP	10.00		16.00

Foreign Currency Transactions Applied to Other Transactions

There are two distinct scenarios that can result in a gain or loss:

- Foreign Currency Transactions Applied to Transactions of the Same Currency, page 3-57
- Cross-Currency Transactions: Transactions Applied to Transactions of a Different Currency, page 3-58

Foreign Currency Transactions Applied to Transactions of the Same Currency

In the case of foreign currency transactions applied to transactions of the same currency, the following values can be different between the transaction and the transactions to which it is applied:

- Conversion Rate
- Conversion Date
- Conversion Rate Type
- Accounted Amount
- Entered Amount

Consider a foreign currency receipt applied to an invoice where the receipt and invoice are in the same currency. If business flow is implemented, the Financial Services Accounting Hub gets the conversion information from the invoice and calculates the accounted amount for the receipt application.

Receivables Example: Assuming Calculate Accounted Amount and Calculate Gain or Loss Amounts are both enabled for the receipt event class, a receipt of GBP 110 is recorded in the system on January 31st with a rate of 1.70 (USD 187). It is applied to and fully pays the GBP 110 invoice recorded on January 18th with a rate of 1.60 (USD 160).

The table below lists source values that should be stored in the transaction object.

Note: For both lines, the receipt line type is Application, the invoice exchange rate is 1.60, and the invoice exchange date is 18-Jan-05.

Receipt Transaction Object Lines

Line No.	Receipt Currency	Receipt Entered Amount	Receipt Exchange Rate	Receipt Exchange Rate Type	Applied to Invoice	Applied to Distribution
1	GBP	132.00	1.7	Corporate	445	3021

Based on a simple application accounting definition, the table below lists the details of the accrual basis subledger journal entry for the receipt.

Accrual Basis: Receipt Applied Subledger Journal Entry

Line No.	Journal Line Type	Entered Currency Code	Entered DR	Entered CR	Exch. Rate	Accounted DR (USD)	Accounted CR (USD)
1. Cash	01.4300.000	GBP	110.00		1.70	187.00	
2. Receivable	01.6000.001	GBP		110.00	1.60		176.00
3. Gain or Loss	01.7000.000	GBP		0	1.70		11.00
					Total	187.00	187.00

The Financial Services Accounting Hub identifies the accounting entries of the invoice based on the applied to information. It then retrieves the currency conversion information, currency code and General Ledger account from the invoice's accounting entry. This information is used for the receipt accounting entries line number 2. Accounted amount is calculated as the receipt's entered amount multiplied by the invoice's conversion rate.

Cross-Currency Transactions: Transactions Applied to Transactions in a Different Currency

This scenario is similar to Foreign Currency Transactions Applied to Transactions of the Same Currency, page 3-57, with the same currency receipt application. The receipt currency is EUR with a conversion rate of 1.4 as shown in the table below.

Receipt Transaction Object Lines

Line Number	Receipt Currency	Receipt Entered Amount	Receipt Exchange Rate	Receipt Exchange Rate Type	Applied to Invoice	Applied to Distribution
1	EUR	132.00	1.4	Corporate	445	3021

Based on a simple application accounting definition with business flows set up, the table below lists the details of the resulting accrual basis subledger journal entry for the receipt.

Accrual Basis: Receipt Applied Subledger Journal Entry

Line No and Journal Line Type	Account	Entered Currency Code	Entered DR	Entered CR	Exch. Rate	Accounted DR (USD)	Accounted CR (USD)
1. Cash	01.4300.000	EUR	132.00		1.40	184.80	
2. Receivable	01.6000.001	GBP		110.00	1.60		176.00
3. Gain or Loss	01.7000.000	EUR		0	1.40		8.80
					Total	184.80	184.80

Reporting Currency Amounts Guideline

If the Financial Services Accounting Hub is configured to calculate reporting currency amount, there is no need to provide any reporting currency information in the transaction objects.

If the Financial Services Accounting Hub is not configured to calculate reporting currency amount and if there is subledger level reporting currency associated with the primary ledger, then the reporting currency amount must be provided in the transaction objects for the subledger application.

Receivables Example: Assume that the Financial Services Accounting Hub is not configured to calculate reporting currency amount for the Receivables application. A ledger is defined with functional currency USD. A subledger level reporting currency CAD is assigned to the USD primary ledger. An invoice of GBP 100 is recorded in the

system on January 18th with a conversion rate of 1.60 to USD and 1.40 to CAD.

The table below lists the source values that are stored in the transaction object lines for each transaction distribution.

Transaction Object Lines

Line No.	Line Type	Entered Currency Code	Entered Amount	Ledger ID	Accounted Amount	Conv. Rate	Conv. Date	Conv. Type
1	Item	GBP	100	25	160*	1.6	18-Jan-05	User
2	Tax	GBP	10	25	16*	1.6	18-Jan-05	User
3	Item	GBP	100	26	140	1.4	18-Jan-05	Reporting
4	Tax	GBP	10	26	14	1.4	18-Jan-05	Reporting

Note: * The Financial Services Accounting Hub calculates this amount if the Calculate Accounted Amounts is enabled for the event class.

Corrections for Rounding Guideline

The types of rounding errors are:

Transaction rounding differences, page 3-60

Journal rounding differences, page 3-60

Transaction Rounding Differences

Transaction rounding differences arise when transaction amounts are distributed to a higher level of detail than the original transaction. In this case, the total accounted amounts for a subledger transaction may not equal the sum of the entered amounts multiplied by the conversion rate. The Financial Services Accounting Hub uses transaction rounding references and rounding class to calculate the transaction rounding difference. This is corrected by adding the difference in the amount into one of the journal lines.

See: Transaction Rounding Reference Accounting Attribute, page 3-38

Journal Rounding Errors

The Financial Services Accounting Hub stores unrounded and rounded entered amount as well as unrounded and rounded accounted amounts on a journal entry. All amounts

in the transaction objects are treated as unrounded amounts. The unrounded amounts on the journal entry lines are rounded based on the precision and minimum accountable unit defined for the currency. If appropriate, an explicit rounding line is created to correct journal rounding differences. This rounding line uses the rounding difference tracking account defined in the ledger.

Only one rounding line is created per unbalanced balancing segment value and entered amount and accounted amount. It is also possible that it debits rounding in entered amount but credits in accounted amount or vice-versa. In this case, the entered amount is negative and shares the same side as the accounted amount.

To illustrate journal rounding errors, the table below lists the details of transaction object lines on an invoice.

Transaction Objects Lines

Line Number	Invoice Line Type	Invoice Entered Currency Code	Invoice Entered Amount
1	Item	USD	10.004
2	Item	USD	10.004

Journal line types Invoice Liability and Expense are used to create journal entries for this invoice. The Merge option is set to Yes for Expense and to No for Invoice Liability.

The table below lists the resulting journal entries before the journal rounding routine is performed.

> **Note:** Entered amount and unrounded entered amount are also stored in the Financial Services Accounting Hub although only accounted amounts are shown in the table below.

Line Number	Account	Unrounded Accounted DR	Unrounded Accounted CR	Accounted DR	Accounted CR
1	Expense	20.008		20.01	
2	Liability		10.004		10.00
3	Liability		10.004		10.00

Since Unrounded Accounted DR equals the total of Unrounded Accounted CR but

Accounted DR does not equal the total of Accounted CR, an explicit journal rounding line is created as described in the table below.

Line Number	Account	Unrounded Accounted DR	Unrounded Accounted CR	Accounted DR	Accounted CR
1	Expense	20.008		20.01	
2	Liability		10.004		10.00
3	Liability		10.004		10.00
4	Rounding		0		0.01

Related Transactions, Lines, and Distributions Guideline

The data from related transactions should be made available as sources. These transactions include applied to transactions, matched with transactions, and created from transactions. This is because transaction amounts in a subledger can be allocated within the originating transaction or across different transactions.

Sources should be stored in transaction objects that are not only associated with the event's underlying transaction but also with its related transactions.

Payables Example: An invoice tax amount is allocated to item or freight amounts for the same invoice. The transaction object should include sources values relating to the tax distribution or allocation as well as the item or freight distributions to which it is allocated.

Receivables Example: A receipt can be applied to one or more invoices. Source values should be stored in the transaction object that relates not only to the receipt but also to the invoices, invoice lines, and distributions to which it is applied.

For receipt amounts allocated to invoice tax lines, sources for the related item lines should also be included in the transaction object. This allows subledger journal entry lines to be created based upon the amount allocated to the invoice or the amount allocated to each line on the invoice.

The Business flow feature ensures that accounting information, such as Accounting Flexfields, is automatically copied from the previous transaction in the business flow. For this feature to work, store the transaction and distribution identifiers from the applied to transaction in the transaction objects.

Independence from Accounting Method Guideline

The transaction objects should be independent from the accounting representation

represented by the subledger accounting methods, which determine how transactions are accounted and represented by subledger journal entries.

Subledger transactions can be subject to more than one accounting representation. For example, the alternative representation to cash basis could be accrual basis. The alternative representation to corporate could be fiscal.

> **Note:** Valuation methods are a way of storing alternative representations directly in the subledger applications. The Financial Services Accounting Hub assumes that each subledger transaction and accounting event is for a single valuation method.
>
> Examples of Oracle applications that support more than one valuation method for a transaction are Assets, through asset books, and Costing, through costing methods. These applications must capture one accounting event for each valuation method.

See: Valuation Methods and the Financial Services Accounting Hub, page 2-24

In addition, users and other groups outside development can also create and modify accounting methods. It is therefore not possible to predetermine how transactions are accounted. The source values that are stored in the transaction object must depend solely on the nature of the transaction and its related setup and not on the seeded subledger accounting methods.

As an example, there must be a complete transaction object with all possible sources for an invoice validated accounting event, even if the user has assigned a cash basis accounting method to the corresponding ledger. This is because an implementer cannot know whether users need to use cash or accrual basis accounting. Providing all the sources as part of the transaction object gives users the flexibility to make this decision.

Balanced Debits and Credits Guideline

Balanced debits and credits should not be stored in the transaction object.

Typically, a transaction amount contributes to two subledger journal entry lines, one debit and one credit. However, multiple accounting representations and specific vertical or geographical accounting requirements can result in a transaction amount contributing to none or more than two subledger journal entry lines.

The total transaction amount is the same no matter how many subledger journal entry lines are generated for it. Therefore, the Financial Services Accounting Hub mandates that balanced debits and credits should not be stored in the transaction object.

Oracle Receivables: Consider a Receivables invoice with a single $100.00 item that is not subject to tax. The invoice results in a single transaction object distribution or allocation being stored in the transaction object. Accounting definitions for the Invoice Completed accounting event determines how to use the $100.00 to create subledger journal entry lines. Typically, the single distribution or allocation of $100.00 is used to create two subledger journal entry lines, Debit to Trade Receivables for $100.00 and

Credit to Revenue for $100.00.

However, there are many examples of requirements for more than two subledger journal entry lines to be generated for a single transaction amount. These include the requirements of the US federal public sector (USSGL Transaction Codes), and requirements for multiple representations (Combined Basis or Corporate and Fiscal). In all these cases, multiple subledger journal entry lines must be created.

In this particular example, the Financial Services Accounting Hub requires that an unbalanced transaction amount of $100 be stored in the transaction object. The subledger accounting method determines the distributions for this transaction amount.

Transaction Objects Lines and Default Accounting Flexfields

The startup application accounting definitions delivered with the application should emulate the current accounting for transactions. Therefore, all Accounting Flexfields used for subledger journal entry lines generated prior to the uptake of the Financial Services Accounting Hub must be stored in the transaction object. This implies that each distribution or allocation must be associated with at least two Accounting Flexfields that correspond to the original debit and credit Accounting Flexfields.

Receivables Example: Consider a receivables invoice with a single $100.00 item that is not subject to tax. This results in a single transaction object line that has both the Trade Receivables and Revenue Accounting Flexfields as sources.

The fact that these Accounting Flexfields are stored on each transaction object line does not imply that there is one subledger journal entry line for each transaction object line. The Financial Services Accounting Hub has features that automatically summarize subledger journal entry lines to the appropriate level of detail so that there is one Trade Receivables subledger journal entry line per invoice.

Specific accounting requirements can result in no subledger journal entry or one with multiple lines being created.

Granularity of Transaction Objects Lines Guideline

For any transaction, the maximum level of detail is stored in the transaction objects and made available to the accounting program. The level of detail refers to all transactions, lines, and distributions to which transaction amounts are allocated.

The transaction objects should normally allocate transaction amounts to the maximum level of detail. This provision affords users greater flexibility in the method of accounting for transaction object amounts.

Since the transaction object has no means of knowing how transaction amounts are accounted, the level of detail of the transaction object cannot be adjusted according to the application accounting definition. The most flexible solution is to allocate transaction amounts to the maximum level of detail.

Receivables Example: An invoice is issued with a single tax amount that is allocated to many invoice distributions. To provide the maximum level of detail, the transaction

object should allocate the tax amount to each of the invoice distributions and make them available to the accounting program.

Providing the maximum level of detail enables users to account for the tax amount either at the level of the tax code and rate or at the level of invoice lines. If the user has elected to account at the tax code level, the accounting program normally creates a single subledger journal entry line for all the allocations or distributions.

Transaction Objects Detail vs. Application-specific Allocation Options

When uptaking the Financial Services Accounting Hub, analyze application setup options that control the level of detail of allocations to determine whether such setup options can be obsoleted by the Financial Services Accounting Hub.

Oracle Example: Some Oracle applications support application and transaction specific options that control how transaction amounts are allocated to each other. For example, in Receivables, the Application Rule Set determines how cash receipt amounts are allocated to invoice lines. In Payables, the Payment Discount Method determines how discount amounts are allocated to invoice lines. Providing the lowest level of detail enables users to either use that detail or summarize it.

In most cases, these options are simply mechanisms that affect the level of detail of allocations. The same results can be obtained by always providing the maximum level of detail and letting the application accounting definitions in the AMB determine how the amounts should be grouped together for accounting purposes. Summarize lines by using the merge matching lines option on the journal line type in the AMB.

Providing the maximum level of detail to the AMB eliminates the need to set up application-specific accounting options to control the level of detail of allocations.

Identifying Distribution or Allocation Amounts in the Transaction Object Detail

To take advantage of the transaction amounts allocated at the maximum level of detail, it should be possible for users to identify and distinguish between different types of transaction object lines. The transaction object must therefore provide sources that describe both the type of transaction amount that has been allocated as well as the type of transaction amount to which it is allocated.

Payables Example: Classify payment amounts into Payment Discount, Gain or Loss, and Rounding. Allocate invoice line distribution amounts to Freight and Item, Recoverable Tax, and Non Recoverable Tax. By storing each of these in the transaction object, users can create a separate subledger journal entry line for each amount.

See: Transaction Objects Example for Payables, page 3-101

Transaction Object Detail: Exceptions to Providing Detailed Allocations

There may be some circumstances under which it is acceptable not to provide the maximum level of detail in an allocation. There are cases where the transaction model does not support allocations to the maximum level of detail. If there is no business case

served by allocating a transaction amount down to the maximum level of detail, you can decide not to store the detailed amounts in the transaction object.

Receivables Example: Receivables does not support the allocation of adjustment amounts to invoice lines. In this case, you can choose to implement a simple allocation rule in the module; for example, by prorating an adjustment amount to all transaction lines. The allocated amounts should be stored in the transaction object.

Third Party Control Accounts Guideline

Third party data should be stored in the transaction object as accounting attributes. Several countries in Europe and Latin America require that subledger systems support third party control account reporting and reconciliation.

Third party control accounts record a subledger's balance or activity in relation to third parties. These third parties are legal entities with which the company engages in business transactions. Examples of third party legal entities are customers and suppliers.

Whenever a subledger transaction that is associated with a customer or supplier is accounted, the Financial Services Accounting Hub records the customer or supplier and site on the journal entry lines for subsequent user analysis and reporting. To accurately account for the subledger transaction and meet all control requirements, third party identifiers must be stored in the transaction objects.

If third party control accounts are enabled for applications, the accounting program performs the following validation:

- If an account derivation rule derives a control account, the accounting program verifies that the account is set in General Ledger to support Supplier, Customer, or Any, and validates that the third party accounting attributes are populated as described in the table below.

Accounting Attribute	Supplier	Customer
Third Party Type	S	C
Party Identifier	AP_SUPPLIERS.VENDOR_ID	HZ_CUST_ACCOUNTS_ALL.CUST_ACCOUNT_ID
Party Site Identifier	AP_SUPPLIER_SITES_ALL.VENDOR_SITE_ID	HZ_CUST_SITE_USES_ALL.SITE_USE_ID

Transactions Associated with Third Parties

For subledger transactions associated with a third party, you should include the values in the table below as standard sources and also map them to the corresponding

accounting attributes.

Standard Source Description	Accounting Attribute Name
Third party internal identifier; for example, party_id or vendor_id	Party Identifier
Third party site internal identifier; for example, party_site_id or vendor_site_id	Party Site Identifier
Third party type (C for customer or S for supplier)	Party Type

See: Accounting Attributes Guideline, page 3-17

Payables Example: An invoice is registered in the system with a supplier and supplier site. The vendor_id or party_id should be stored as the party identifier; the vendor_site_id or party_site_id should be stored as the party site identifier; and S should be stored as the party type. The standard sources containing this information should be assigned to the corresponding accounting attributes.

Receivables Example: A receipt is registered in the system with a customer but no customer site. The party_id for the customer should be stored as the party identifier; the party site identifier should be left blank; and C should be stored as the party type.

Applied to Transactions Associated with Third Parties

The business flow feature ensures that accounting information, such as Accounting Flexfields, are automatically copied from the previous transaction in the business flow.

Subledger Journal Entry Creation

If the Create Accounting program finds a subledger journal entry line with a third party control account, it calls an API to check the following:

- The subledger associated with the third party control account and the subledger of the journal entry must be the same.
- The third party information must not be null.

If an error is returned, the Create Accounting program does not create a subledger journal entry.

Subledger Accounting Balances Updates

The Subledger Accounting Balances Updates program is generated by the Create Accounting program when final journal entries are created and maintains balances for third party control accounts.

Related Topics

Third Party Control Accounting Attributes, page 3-32

Defining Segment Qualifiers, *Oracle General Ledger User Guide*

Translated Sources, Lookup Types, and Value Sets Guideline

The transaction objects should store a translated source value in each installed language that is used by ledgers linked to the accounting event's primary ledger. The accounting program creates subledger journal entry descriptions and supporting references using the language that is selected as part of the ledger setup.

You can determine what languages are required to process a primary ledger and its accounting events.

See: SQL Query to Determine Languages Used, page 3-69

Translated Sources

When analyzing transaction object requirements, determine whether the column value on which a source is based is translatable. If it is, then the source should be defined as translatable in the AMB.

Sources that are translated cannot be used in AMB conditions.

See: Account Derivation Rule Conditions, *Oracle Subledger Accounting Implementation Guide*

Lookup Types

If a source is a lookup code, the Financial Services Accounting Hub displays its corresponding meaning in the appropriate language when it is used in header and line descriptions or supporting references. Lookup types can:

- Reduce the number of source values that need to be stored in the transaction objects

  As part of the definition of standard sources, you can optionally specify a lookup type. If a lookup code is defined as a source that is assigned a lookup type, there is no need to define a source for the corresponding meaning. The Financial Services Accounting Hub converts the lookup code into the appropriate and translated meaning whenever it is displayed to users.

- Help prevent errors by displaying valid, user friendly names for sources that are lookup codes

  Users select from the list of valid values.

- Overcome many of the restrictions associated with translated sources

  When defining conditions in the AMB, the Financial Services Accounting Hub

displays the translated meaning to users and stores the untranslated lookup code. Since the untranslated lookup code is used as the condition, conditions function independently of the language of the user or the ledger.

Payables Example: The transaction object should store the Invoice Line Type lookup code and not the corresponding (translated) meaning.

Value Sets

Value sets provide features similar to lookup types for lists of values not stored as lookups. Instead of a lookup type, the value set derives the user meaning corresponding to a hidden identifier, such as a surrogate key.

The disadvantage of using a value set instead of a lookup type is that it requires more setup; you should define a value set to link the hidden identifier to its user friendly and translated meaning.

Optionally, specify a value set as part of the definition of standard sources. If a hidden identifier, such as a surrogate key, is defined as a source that is assigned a value set, there is no need to define a source for the corresponding meaning. The Financial Services Accounting Hub converts the hidden identifier into the appropriate and translated meaning whenever it is displayed to users.

See: Assign Sources, page 5-9

The value sets associated with sources should not rely on contexts such as operating unit or asset book to return a unique value. For example, it is not possible to use a value set for tables that are partitioned by operating unit. If the value set uses a secure view, no values are displayed because the accounting program does not run within the context of a single operating unit. If the value set uses the base table, it is possible that duplicate values will be returned. Therefore, value sets should not be used if the value set cannot return a list of unique values without recourse to another explicit parameter.

Displayed Name of Sources that have been Assigned Lookup Types or Value Sets

The displayed name of sources that have been assigned lookup types or value sets should describe the source's meaning. Although the transaction object contains only a lookup code or value set ID, the source's displayed name should describe the corresponding meaning.

Payables Example: Create a source with the display name Invoice Type. Assign a lookup type of INVOICE_TYPE to this source. The transaction object contains the invoice type lookup code.

SQL Query to Determine Languages Used

The following sql statement returns a list of the languages used to generate subledger journal entries for the subledger application's primary and secondary ledgers. The values returned, language_code, corresponds to fnd_languages.language_code.

The sql accepts the subledger application identifier and the primary ledger identifier as

parameters.

```
select distinct nvl(led.sla_description_language, userenv('LANG'))
language_code
from  gl_ledgers led, xla_ledger_options lopt
where led.ledger_category_code in ('PRIMARY', 'SECONDARY')
and  led.ledger_id = lopt.ledger_id and lopt.enabled_flag = 'Y'
and  lopt.application_id = &subledger_application_id
and  led.primary_ledger_id = &primary_ledger_id;
```

The transaction object program should populate the transaction objects with the translated source values for the languages codes. The language codes are the same for all accounting events that belong to a primary ledger.

Consistency Guideline

Ensure that sources stored in the transaction objects are stored in a consistent manner across different types of transactions.

For example, if the source Purchase Order Number is made available for invoice lines that are matched to a purchase order line, then the Purchase Order Number must be stored whenever an invoice line is matched to a purchase order number.

Sources corresponding to the same underlying transaction values should be defined only once and assigned to all event classes using them.

Receivables Example: A Transaction Type source should be defined once and assigned to both the Invoice and Receipt Application event classes.

It is not correct to define separate Invoice Transaction Type and Applied To Transaction Type sources for the underlying transaction type and assign them separately to the Invoice and Receipt Application event classes.

This guideline is important because users can share journal entry setups, such as account derivation rules and descriptions, across event classes if they use the same sources. If separate sources are defined for each event class or event entity, separate account derivation rules and descriptions must be defined for each event entity or class. This results in increased complexity, maintenance, and inconsistencies.

Receivables Example: Assume that separate sources are defined for the underlying transaction type and assigned separately to the Invoice and Receipt Application event classes. In this case, it is not possible to share a Receivable account derivation rule between the Invoice and Receipt Application event classes even though the account derivation rule is exactly the same.

Tax References Guideline

The transaction objects should store these tax references as accounting attributes. Tax references establish a link between information stored in the tax repository and subledger journal entries. Tax reports and inquiries use the link to provide a complete

reconciliation of tax amounts and accounting.

When calling tax routines to calculate the tax amounts for their transactions, the routines return one or more of the following tax references:

- Detail Tax Line Reference
- Summary Tax Line Reference
- Detail Tax Distribution Reference

The transaction objects should store these tax references as accounting attributes. If the tax references are not correctly stored in the transaction objects, it is not be possible to reconcile subledger journal entry lines with tax amounts in the tax repository.

> **Note:** The accounting attribute assignment for tax group accounting attributes will not enforce mapping of all related attributes.

When the accounting program inserts records into the distribution link table, values of the tax references are stored along with the distribution identifiers.

Flexfields Guideline

The transaction objects should store flexfield column values as sources if they can be used in creating subledger journal entries. This guideline is only applicable if Oracle subledger applications are used.

Accounting Flexfields

All Accounting Flexfields that may be related to a transaction distribution or entered as part of the default setup should be stored in the transaction objects as sources.

It is necessary to store only the code combination identifier of the Accounting Flexfield in the transaction objects. The Financial Services Accounting Hub can retrieve individual segments values of the flexfield directly from the code combinations table.

Key Flexfield Segments

Wherever a key flexfield, other than an Accounting Flexfield, stores important information about a transaction or distribution, the key flexfield code combination identifier should be stored in the transaction objects. Individual segment values do not need to be stored, as the Financial Services Accounting Hub can retrieve them from the underlying code combinations table. For example, users working with the Assets subledger application store information in the Asset Category flexfield. They may want to create application accounting definitions based on the value of an individual segment in this flexfield. Therefore, the Asset Category Flexfield Combination identifier should be stored in the transaction objects.

In cases where applications provide features that enable users to define multiple key

flexfields and assign them directly or indirectly to transactions, decide which values should be stored in the transaction object. For example, Costing may want to store both the default item category set and the accounting category set.

Descriptive Flexfield Attributes

Not all descriptive flexfield attributes need to be stored in the transaction objects. At a minimum, the Financial Services Accounting Hub recommends that you define those descriptive flexfields associated directly with the transaction header and lines as sources. For each descriptive flexfield, the context column must be included in the transaction objects. Include other descriptive flexfields that you think are commonly used.

Global Descriptive Flexfield Attributes

Only global descriptive flexfields that the Global Financial Applications Technical Reference Manual indicates are in use should be stored in the transaction objects.

Oracle Applications strategy is to incorporate the globalization features into the base applications. However, to the extent that this incorporation is in progress, some global descriptive flexfields may need to be stored in the transaction objects.

The following sql statement can be used to identify global descriptive flexfields that are in use:

```
select  distinct f.APPLICATION_TABLE_NAME,c.APPLICATION_COLUMN_NAME
from   fnd_descr_flex_column_usages c,
fnd_descriptive_flexs_vl f
where   c.application_column_name like 'GLOBAL_ATTRIBUTE%'
and    f.DESCRIPTIVE_FLEXFIELD_NAME = c.DESCRIPTIVE_FLEXFIELD_NAME
order by f.APPLICATION_TABLE_NAME, c.APPLICATION_COLUMN_NAME;
```

Accounting Flexfield Derivation Mechanisms Guideline

All values used to derive accounts prior to the uptake of the Financial Services Accounting Hub should be made available as sources.

In prior releases of the Oracle E-Business Suite, applications used a variety of mechanisms to provide flexibility in Accounting Flexfield derivation. These include Receivables AutoAccounting and Projects AutoAccounting, Assets Flexbuilder, Workflow Account Generator, and OPM's MAC. For these solutions to be replaced by the Financial Services Accounting Hub, the values available for creating entries with these mechanisms must be stored in the transaction objects.

Examples of the types of values used for creating accounting in prior releases of the Oracle E-Business Suite include the following:

Assets: Asset Category Accounts, Asset Assignment Accounts, and Asset Book Control Accounts

Projects: Expenditure Types, Service Codes, and Class Codes

Purchasing: Deliver to Location, Deliver to Subinventory, and Expenditure Category

Receivables: Transaction Type, Customers, and Sales Representatives

Accounting Options Guideline

Existing setup accounting options should be stored in the transaction objects as sources. The journal entry setups that make up the startup data application accounting definitions use these sources to determine how to account for subledger transactions.

If users require more than one accounting representation, each based on a different value of an accounting option, they can use the AMB to create application accounting definitions. These definitions implement journal entry setups that correspond to a specific value of an accounting option.

Oracle Payables: Values for the Use Future Dated Payment Account accounting option should be stored in the transaction objects. The account derivation rules for the future dated payment account can take this source value into account when deriving the liability account from the supplier site or payment document.

For accounting options that have a significant impact on accounting, consider creating a dedicated application accounting definition.

The following are examples of accounting options that should be implemented as separate application accounting definitions:

- Cash Basis
- Accrual Basis
- Southern European Permanent Inventory
- Southern European Non Permanent Inventory

Internal Identifiers Guideline

Internal Identifiers

Internal identifiers, such as surrogate keys, should not be included as sources. Internal identifiers typically have no business value and will not be understood by users.

Oracle Receivables: The Transaction Type Name, not the Transaction Type Id should be stored in the transaction objects.

Note the following exceptions:

- Distribution identifiers, a minimum set of surrogate keys, should be stored in the transaction objects.

See: Distribution Identifiers Guideline, page 3-48

- Translated sources and value sets (surrogate keys or internal codes) are stored in the transaction objects but not displayed to users.

  See: Translated Sources, Lookup Types, and Value Sets Guideline, page 3-68

- The code combination identifiers of key flexfields and Accounting Flexfields are stored in the transaction objects but not displayed to users.

  See: Flexfields Guideline, page 3-71

Processing Statuses

Internal processing statuses such as posting status flags or enabled or disabled flags should not be stored in transaction objects.

Payables Example: The accrual posted flag should not be included as a source.

Who Columns

Who columns should not be included as sources. The names of the application's user who created the accounting event and who submitted the accounting program are available as system sources.

System Sources Guideline

System sources are independently populated by the Financial Services Accounting Hub. They should not be stored in transaction objects.

See: Standard, System, and Custom Sources, page 3-6

The table below shows a list of system sources supported by the Financial Services Accounting Hub.

System Sources Supported by the Financial Services Accounting Hub

Name	Data Type	Description
Accounting Event Date	Date	Accounting event date
Accounting Event Creator	Alphanumeric	User who created or updated the subledger transaction at the time that the accounting event was captured
Accounting Event Type Code	Alphanumeric	Accounting event type code

Name	Data Type	Description
Accounting Event Type Name	Alphanumeric	Accounting event type name
Accounting Event Class Code	Alphanumeric	Accounting event class code
Accounting Event Class Name	Alphanumeric	Accounting event class name
Journal Entry Creation Date	Date	Journal entry creation date (system date)
Journal Entry Creator	Alphanumeric	User who submitted the accounting program
Journal Entry Source User Name	Alphanumeric	User who entered journal entry source
First Day Next Accounting Period	Date	Used in accrual reversal to indicate that the first day in the next GL period will be used to create the accrual reversal entry. The day can be rolled forward to the next business day only if the GL transaction calendar is used.
Last Day of Next Accounting Period	Date	Used in accrual reversal to indicate that the last day in the next GL period will be used to create the accrual reversal entry. The day can be rolled backward to the last business day only if the GL transaction calendar is used.
Ledger Name	Alphanumeric	Primary or secondary ledger name associated with the journal entry
Transaction Legal Entity Identifier	Alphanumeric	Transaction legal entity identifier

Name	Data Type	Description
Next Day	Date	Used in accrual reversal to indicate that the next day will be used to create the accrual reversal entry. The day can be rolled forward to the last business day only if the GL transaction calendar is used.
Accounting Period Name	Alphanumeric	Period name associated with the journal entry
Accounting Period Type	Alphanumeric	Period type associated with the journal entry
Application Accounting Definition Name	Alphanumeric	Application accounting definition associated with the journal entry
Application Accounting Definition Owner	Alphanumeric	Indicates whether the application accounting definition is owned by Oracle or User
Application Accounting Definition Version	Alphanumeric	Application accounting method version associated with the journal entry
Financial Services Accounting Hub Method Name	Alphanumeric	Subledger accounting method name
Ledger Currency Code	Alphanumeric	Primary or secondary ledger currency code associated with the journal entry; not reporting currency code
Ledger Language	Alphanumeric	Determines the text format in the description language; ISO language code for the description language
Ledger Language for Dates and Number in Description	Alphanumeric	Determines the format of the number and dates in the description

Name	Data Type	Description
Event Numeric Reference 1	Number	Accounting event reference information captured on the accounting event
Event Numeric Reference 2	Number	Accounting event reference information captured on the accounting event
Event Numeric Reference 3	Number	Accounting event reference information captured on the accounting event
Event Numeric Reference 4	Number	Accounting event reference information captured on the accounting event
Event Alphanumeric Reference 1	Alphanumeric	Accounting event reference information captured on the accounting event
Event Alphanumeric Reference 2	Alphanumeric	Accounting event reference information captured on the accounting event
Event Alphanumeric Reference 3	Alphanumeric	Accounting event reference information captured on the accounting event
Event Alphanumeric Reference 4	Alphanumeric	Accounting event reference information captured on the accounting event
Event Date Reference 1	Date	Accounting event reference information captured on the accounting event
Event Date Reference 2	Date	Accounting event reference information captured on the accounting event
Event Date Reference 3	Date	Accounting event reference information captured on the accounting event

Name	Data Type	Description
Event Date Reference 4	Date	Accounting event reference information captured on the accounting event

Accounting Reversals Guideline

To invoke an accounting reversal for a transaction or transaction distribution, the transaction objects should include the appropriate header or line level accounting reversal options.

Accounting Reversals

Accounting reversals enables users to reverse the accounting impact of a previously accounted transaction distribution or all existing accounting for a transaction. Enable this functionality by implementing the accounting reversals setups described in this section.

The Financial Services Accounting Hub uses the following accounting reversal specific terminology.

Reversed (original) Distribution. Refers to a transaction distribution that although successfully accounted, is either incorrect or canceled. The transaction distribution is therefore reversed.

Reversal Distribution. Refers to a transaction distribution which reverses the effect of the original distribution on transaction balances. Typically, reversal distributions are identical to the reversed distributions in all respects except for entered (ledger) amounts that reverse the sign of the original.

Replacement Distribution. Refers to a transaction distribution which replaces the reversed distribution with the correct value

The table below illustrates the distributions described above.

Distributions Example

Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
1	Item	10-Jan-2006	1000	Reversed

Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
2	Item	12-Jan-2006	-1000	Reversal (of line 1)
3	Item	12-Jan-2006	2000	Replacement (of line 1)
Transaction Total			2000	

Note that the original accounting impact of the reversed distributions is undone, even if the subledger journal setups or accounting configurations have changed since the original subledger journal entry was generated.

Accrual Reversals

Accrual reversal enables users to define how and when accrual reversals are automatically performed . To schedule the automatic reversal of a journal entry at the time it is created, users assign the Accrual Reversal GL Date accounting attribute at the event class level in the Accounting Attributes Assignments window. Users can override this value in the journal entry header in the Application Accounting Definitions window. This value determines the GL date of the accrual reversal entry. The Financial Services Accounting Hub creates the accrual reversal entry to negate the impact of the accrual entry. Depending on the selection for Reversal Method for the ledger accounting options in the Update Accounting Options window, the accrual reversal will have the debit and credit signs reversed or it will have negative amounts.

See:

- GL Date Accounting Attribute, page 3-31
- Setting up Subledger Accounting Options, *Oracle Subledger Accounting Implementation Guide*

Accrual Reversal GL Date Example

A company receives material worth $100 on Monday, January 12, 2006 that has not been invoiced. To book the accrual, the Accrual Expense account is debited $100.00 and the Accrual Liability account is credited $100.00 as shown in the example below.

GL Date: 12-Jan-2006

Account	Entered DR	Credited DR	Accounted DR (USD)	Accounted CR (USD)
Accrual Expense	100.00	-	100.00	-
Accrual Liability	-	100.00	-	100.00

If the Accrual Reversal GL Date is Next Day and the Reversal Method for the ledger and subledger application is Switch DR/CR, the journal entry in the table below is created to reverse the accrual.

GL Date: 13-Jan-2006

Account	Entered DR	Credited DR	Accounted DR (USD)	Accounted CR (USD)
Accrual Liability	100.00	-	100.00	-
Accrual Expense	-	100.00	-	100.00

If the accrual reversal GL Date is First Day Next GL Period and the Reversal Method for the ledger and subledger application is Change Signs, then the following journal entry is created to reverse the accrual.

GL Date: 01-Feb-2006

Account	Entered DR	Credited DR	Accounted DR (USD)	Accounted CR (USD)
Accrual Expense	(100.00)	-	(100.00)	-
Accrual Liability	-	(100.00)	-	(100.00)

If the accrual reversal GL Date is Last Day Next GL Period and the Reversal Method for the ledger and subledger application is Change Signs, then the following journal entry is created to reverse the accrual.

GL Date: 28-Feb-2006

Account	Entered DR	Credited DR	Accounted DR (USD)	Accounted CR (USD)
Accrual Expense	(100.00)	-	(100.00)	-
Accrual Liability	-	(100.00)	-	(100.00)

See: Accrual Reversals, *Oracle Subledger Accounting Implementation Guide*

Line Accounting Reversals

Implementation of the accounting reversal functionality in the Financial Services Accounting Hub is discussed in the following sections:

- Transaction Objects, page 3-81
- Accounting Program, page 3-82

Transaction Objects

To successfully invoke a line accounting reversal for a transaction distribution, the information stored in the transaction objects is different from that of a standard transaction object line.

To reverse the accounting impact of the original distribution, the accounting program must have access to the identifiers that were stored in the transaction objects when the reversed (original) distribution was first accounted. For all reversal distributions, the accounting program stores the reversed (original) distribution identifiers along with the reversal distribution identifiers in the distribution links table.

Accounting attributes enable you to pass the accounting reversals options to the accounting program. Assign sources to these accounting attributes for all event classes that you want to enable accounting reversals for.

For reversal distributions, the transaction object line must provide values for the following accounting attributes:

- The Accounting Reversal Option must contain the value Y.
- Distribution identifiers of the reversed (original) distributions:
  - Reversed Distribution Link Type
  - Reversed Distribution Identifiers
- Distribution identifiers of the reversal (new) distributions:

- Distribution Link Type
- Distribution Identifiers

It is not necessary for the transaction objects to store any other line level sources except if standard accounting is required for the reversal distribution in addition to the line accounting reversal. In this case, the source mapped to the accounting attribute Line Accounting Reversal Option must contain the value B.

The processing of line accounting reversals does not affect the header level transaction objects. Users can use header level sources to create the subledger journal entry header description.

Accounting Program

When the accounting program encounters a line level transaction object with the Line Accounting Reversal Option set to Y, it does the following:

- Uses the Reversed Distribution Identifiers to identify all subledger journal entry lines created by the reversed transaction distributions
- Generates subledger journal entry lines that reverse the accounting impact of the reversed (original) distributions

  These lines are created with information in tables as follows:

  - Derive entered, base, and statistical amounts from the distribution link table.
  - Derive Accounting Flexfield, accounting class, third party identifiers, description, and conversion rate information from the subledger journal entry lines table.

- Creates distribution links that link the reversal (new) distributions with the new subledger journal entry lines
- Creates standard accounting for the transaction distribution in addition to the line accounting reversal if the line Accounting Reversal indicator contains the value B

Uniqueness of Distribution Identifiers.

The accounting program does not enforce the uniqueness of distribution identifiers. Distribution identifiers must uniquely identify the transaction distribution or the accounting feature does not work.

Transaction Accounting Reversals

Populate the accounting attribute header Transaction Accounting Reversal Indicator with the value Y. The accounting program reverses all existing accounting for the transaction associated with the accounting event. Line extract objects do not need to be populated.

See: Transaction Accounting Reversal Accounting Attribute, page 3-30

Accounting Reversal Examples

The examples in this section illustrate the mechanism for accounting reversals. They are analyzed through the use of the following tables:

- Invoice Header (owned by the Financial Services Accounting Hub)
- Invoice Distribution (owned by the Financial Services Accounting Hub)
- Transaction Objects (owned by subledger applications)
- Subledger Journal Entry Lines (owned by subledger applications)
- Distribution Link Table Entries (owned by subledger applications)

Assume that the event entity Invoice in the AMB has the First Distribution Identifier accounting attribute assigned to the Invoice Distribution Identifier standard source.

Example 1: Line Accounting Reversal Only

This section includes the following parts:

- Invoice Validated, page 3-83
- Invoice Canceled, page 3-85

Invoice Validated

Consider an invoice that is entered into the system and validated on 10-Jan-2006. The table below lists the details of the invoice header.

Invoice Header

Supplier Name/Number	Invoice Type	Invoice Number
ABC Ltd./XYZ	Invoice	A-1234

The table below lists the details of the invoice distribution.

Invoice Distribution (Validated)

Invoice Distribution Identifier	Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
1234	1	Item	10-Jan-2006	1000	Standard Item
1235	2	Tax	10-Jan-2006	200	VAT @20%

The table below shows the simplified line level transaction object that stores the information for this distribution.

Invoice Validated Transaction Objects Lines

Invoice Distribution Line Number	Accounting Reversal Option	Invoice Distribution Identifier	Invoice Line Type	Invoice Entered Amount	Description	First Reversed Distribution Identifier
1	N	1234	Item	1000	Standard Item	
2	N	1235	Tax	200	VAT @20%	

The table below shows the subledger journal entry lines that are created for the invoice.

Invoice Validated Subledger Journal Entry Lines

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
6000	1	101	Liability	01.4300.000		1200	Lines 1 + 2
6000	2	101	Expense	01.6000.001	1000		Line 1
6000	3	101	Tax	01.6001.000	200		Line 2

The following subledger accounting distribution link table records how each of the transaction object amounts contributed to the subledger journal entry lines.

Financial Services Accounting Hub Distribution Link Table Lines

Entry Header Identifier	Entry Line Number	Ledger ID	First Distribution Identifier	Entered Amount	Ledger Amount	Accounting Reversal Option
6000	1	101	1234	1000	1000	N
6000	1	101	1235	200	200	N
6000	2	101	1236	1000	1000	N
6000	3	101	1237	200	200	N

Invoice Canceled

Assume the user cancels the invoice on 20th January, 2006. A new accounting event and invoice distributions are created for the invoice cancellation. The table below lists the details of the invoice header of this invoice.

Invoice Header (Canceled)

Supplier Name/Number	Invoice Type	Invoice Number
ABC Ltd./XYZ	Invoice	A-1234

The table below lists the details of the original (distribution identifiers 1234 and 1235) and new (distribution identifiers 1400 and 1401) distribution lines.

Invoice Distribution (Canceled)

Invoice Distribution Identifier	Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
1234	1	Item	10-Jan-2006	1000	Standard Item
1235	2	Tax	10-Jan-2006	200	VAT @20%
1400	3	Item	20-Jan-2006	-1000	Cancels Line 1

Invoice Distribution Identifier	Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
1401	4	Tax	20-Jan-2006	-200	Cancels Line 2

Assuming that an accounting reversal is required, the line level transaction object should store the following:

- For the lines that complete the reversal, set the transaction object line column Accounting Reversal Option to Y.
- Populate the reversed distribution identifier columns with the distribution identifiers of the reversed (original) transaction distributions.
- Populate the standard sources that are mapped to the distribution identifier accounting attributes with the distribution identifiers of the reversal transaction distributions.

It is not necessary for the transaction object to store any other source values. The table below shows the line level transaction object for the lines that complete the reversal.

Invoice Canceled Transaction Objects Lines

Invoice Distribution Line Number	Accounting Reversal Option	Invoice Distribution Identifier	Invoice Line Type	Invoice Entered Amount	Comment	First Reversed Distribution Identifier
3	Y	1400	No value required	No value required		1234
4	Y	1401	No value required	No value required		1235

The Financial Services Accounting Hub takes account of the ledger level Reverse Side or Sign user preference when determining the side and sign of the subledger journal entry line amounts. The preference is set for a ledger within the context of an application.

In this example, assume that the preference is set to Reverse Sign. The table below shows the subledger journal entry lines for the cancellation.

Invoice Canceled Subledger Journal Entry Lines

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
20000	1	101	Liability	01.4300.000		-1200	Lines 3 + 4
20000	2	101	Expense	01.6000.001	-1000		Line 3
20000	3	101	Tax	01.4372.000	-200		Line 4

The corresponding distribution link table lines are shown in the table below.

Financial Services Accounting Hub Distribution Link Table Lines

Entry Header Identifier	Entry Line Number	Ledger ID	First Distribution Identifier	Entered Amount	Ledger Amount	Accounting Reversal Option
20000	1	101	1400	-1000	-1000	Y
20000	1	101	1401	-200	-200	Y
20000	2	101	1400	-1000	-1000	Y
20000	3	101	1401	-200	-200	Y

Example 2: Line Accounting Reversal and Standard Accounting, Separate Distribution

Assume that instead of canceling the invoice as in the prior example, the user adjusts one of the invoice distributions. The table below lists the header details of the adjusted invoice.

Invoice Header (Adjusted)

Supplier Name/Number	Invoice Type	Invoice Number
ABC Ltd./XYZ	Invoice	A-1234

The table below lists the distribution lines of the invoice.

Invoice Distribution (Adjusted)

Invoice Distribution Identifier	Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
1234	1	Item	10-Jan-2006	1000	Standard Item
1235	2	Tax	10-Jan-2006	200	VAT @20%
1600	3	Item	20-Jan-2006	-1000	Reverses Line 1
1601	4	Freight	20-Jan-2006	1000	Replaces Line 1

Assume that an accounting reversal is required to account for the reversal of the item line. The transaction objects contain an accounting reversal transaction object line for the reversal line and a standard transaction object line for the replacement distribution. The table below shows the transaction object lines of the adjusted invoice.

Invoice Adjusted Transaction Objects Lines

Invoice Distribution Line Number	Accounting Reversal Option	Invoice Distribution Identifier	Invoice Line Type	Invoice Entered Amount	Comment	First Reversed Distribution Identifier
3	Y	1600				1234
4	N	1601	Freight	1000	Replaces Line 1	

Assume that the application accounting definitions applied to the replacement distribution create subledger journal entry lines with the following Accounting Flexfields:

- Liability account that is the same as the original
- Expense account that is different from the original

The table below shows the subledger journal entry lines for the cancellation.

Invoice Adjusted Subledger Journal Entry Lines

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
30000	1	101	Expense	01.6000.001	-1000		Accounting Reversal
30000	2	101	Expense	01.6000.000	1000		Standard Accounting
30000	3	101	Liability	01.4300.000		-1000	Accounting Reversal
30000	4	101	Liability	01.4300.000		1000	Standard Accounting

Since the liability lines cancel each other out, the Financial Services Accounting Hub eliminates them, leaving a subledger journal entry with two subledger journal entry lines.

The table below shows the corresponding distribution link table lines.

Financial Services Accounting Hub Distribution Link Table Lines

Entry Header Identifier	Entry Line Number	Ledger ID	First Distribution Identifier	Entered Amount	Ledger Amount	Accounting Reversal Option
30000	1	101	1600	-1000	-1000	Y
30000	2	101	1601	1000	1000	N
30000	3	101	1600	-1000	-1000	Y
30000	4	101	1601	1000	1000	N

Example 3: Standard Accounting and Line Accounting Reversal, Same Distribution

The previous accounting reversal example assumes that each time the user modifies a transaction that has already been accounted, the application creates new reversal and replacement transaction distributions.

However, it is not necessary to create new reversal and replacement transaction distributions to correctly process accounting reversals. The transaction object that creates the accounting reversal and replacement can use the same distribution link identifiers as the original transaction distribution.

This example assumes a bond that is mark to market (MTM) at the end of each business day.

> **Note:** Mark-to-market is the act of recording the price of a security on a daily basis to calculate profits and losses or to confirm that margin requirements are being met; it reflects the current market value and not the book value.

The scenario is described below and in the following table:

- 01/18/06: Bond Inception. The original bond price is $100.
- 01/19/06: Bond market price is $105. An MTM of $5 is booked to adjust the bond price.
- 01/20/06: Bond market price is $103. The previous MTM is reversed and a new MTM of $3 is booked.

Transaction Objects Lines for the Bond Inception, MTM, Reversal and Creation of New MTM

Event Type	Event ID	Distribution ID	GL Date	Amount	Accounting Reversal Indicator	Reversal Distribution ID
Bond Inception	102	1001	18-JAN-2006	100		
Bond MTM	103	1002	19-JAN-2006	5		
Bond MTM	104	1003	20-JAN-2006	3	B	1002

The table below describes the subledger journal entry lines for the bond inception with a GL Date of 18-JAN-2006.

Subledger Journal Entry Lines, 18-JAN-2006

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
30001	1	101	Asset	01.7010007.000	100		Standard Accounting
30001	2	101	Cash	01.7017002.000		100	Standard Accounting

In the above table, the total accounted amount is $100 for Debit and $100 for Credit.

The table below describes the subledger journal entry lines for the Bond MTM with a GL date of 19-JAN-2006.

Subledger Journal Entry Lines, 19-JAN-2006

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
30002	1	101	Asset	01.7010007.000	5		Standard Accounting
30002	2	101	Unrealized Gain	01.7017002.000		5	Standard Accounting

In the above table, the total accounted amount is $5 for Debit and $5 for Credit.

The Financial Services Accounting Hub takes account of the ledger level Reverse Side or Sign user preference when determining the side and sign of the subledger journal entry line amounts. The preference is set for a ledger within the context of an application.

In this example, it is assumed that the preference has been set to Switch Sides. The table below describes the subledger journal entry lines for the reversal of the previous MTM and the creation of the new MTM with a GL date of 20-JAN-2006.

Subledger Journal Entry Lines, 20-JAN-2006

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
30003	1	101	Asset	01.7010007.000		5	Accounting Reversal
30003	2	101	Unrealized Gain	01.7017002.000	5		Accounting Reversal
30003	3	101	Asset	01.7010007.000	3		Standard Accounting
30003	4	101	Unrealized Gain	01.7017002.000		3	Standard Accounting

For Account 701000, $5 is credited for the reversal of the original MTM and $3 is debited for the booking of the new MTM.

For Account 7017002, $5 is debited for the reversal of the original MTM and $3 is credited for the booking of the new MTM.

The table below shows the corresponding distribution link table lines for the Accounting Reversal Entry.

Financial Services Accounting Hub Distribution Link Table Lines

Entry Header Identifier	Entry Line Number	Ledger ID	First Distribution Identifier	Entered Amount	Ledger Amount	Accounting Reversal Option
30003	1	101	1003	5	5	Y
30003	2	101	1003	5	5	Y
30003	3	101	1003	3	3	N
30003	4	101	1003	3	3	N

Example 4: Transaction Accounting Reversal

The following is an example of a header level transaction accounting reversal

(Transaction Accounting Reversal = Y).

The scenario is described below and in the following table:

- 01/09/06: Bond is purchased (Bond Inception) with an issue price of $1000 and a premium of $100.
- 01/10/06: The entire transaction is reversed (Bond Reversal). The corresponding extract header is marked as a transaction accounting reversal.

Extract Lines for Both the Bond Purchase and Premium and Extract Header for Bond Transaction Reversal

Event Type	Event Identifier	Distribution Identifier	GL Date	Amount	Transaction Accounting Reversal
Bond Inception	1000	1	09-Jan-2006	1000	
Bond Inception	1000	2	09-Jan-2006	100	
Bond Inception	1001		19-Jan-2006		Y

Note: Notice that both the Accounting Reversal Indicator and Reversal Distribution Identifier are Null.

Subledger Journal Entry Lines for the Bond Inception with a GL Date of 09-JAN-2006

Event Type	Line Number	Account	Debit	Credit
Bond Inception	1	9100000	1000	
Bond Inception	2	9100020	100	
Bond Inception	3	2010000		1100

The Total Accounted Amounts for the above table are as follows:

- Debit = 1100
- Credit = 1100

Subledger Journal Entry Lines for the Bond Reversal with a GL Date of 10-JAN-2006

Event Type	Line Number	Account	Debit	Credit
Bond Reversal	1	9100000		1000
Bond Reversal	2	9100020		100
Bond Reversal	3	2010000	1100	

- Debit = 1100
- Credit = 1100

The Distribution Links table (XLA_DISTRIBUTION_LINKS) below displays the link between the distributions on the transaction and the resulting journal entries.

Event Type	Temporary Line Number	Event Identifier	Journal Entry Header Identifier	Journal Entry Line Number	Distribution Identifier	Amount	Reference Journal Entry Header Identifier	Reversed Journal Entry Line Number
Bond Inception	1	1000	2000	1	1	1000	2000	
Bond Inception	2	1000	2000	2	2	100	2000	
Bond Inception	3	1000	2000	3	1	1000	2000	
Bond Inception	4	1000	2000	3	2	100	2000	
Bond Reversal	-1	1001	2001	1		1000	2000	1
Bond Inception	-2	1001	2001	2		100	2000	2

Event Type	Temporary Line Number	Event Identifier	Journal Entry Header Identifier	Journal Entry Line Number	Distribution Identifier	Amount	Reference Journal Entry Header Identifier	Reversed Journal Entry Line Number
Bond Inception	-3	1001	2001	3		1000	2000	3
Bond Inception	-4	1001	2001	3		100	2000	3

The first two lines in the above table represent the creation of the two original debit subledger journal entry lines for the Bond Inception.

The third and fourth lines in the above table represent the creation of the one original credit subledger journal entry line for the Bond Inception. The distribution lines are added and summarized in the subledger journal entry line.

The fifth and sixth lines in the above table represent the reversal of the two debit subledger journal entry lines for the Bond Inception with a Journal Entry Header Identifier of 2000 and Journal Entry Line Numbers of 1 and 2 respectively. The Temporary Line Numbers are multiplied by -1 to reflect the reversal of the original lines.

The seventh and eighth lines in the above table represent the reversal of the one summarized credit subledger journal entry line for the Bond Inception with a Journal Entry Header Identifier of 2000 and a Journal Entry Line Number of 3. The Temporary Line Numbers are multiplied by -1 to reflect the reversal of the original lines.

Upgrade Case Scenario for Accounting Reversals

The upgrade case arises when a transaction distribution that has been accounted prior to the uptake of the Financial Services Accounting Hub is subject to an accounting reversal after the uptake of the Financial Services Accounting Hub.

Overview

Accounting reversal relies on distribution links to identify the prior accounting impact of transaction distributions. However, transactions accounted prior to the uptake of the Financial Services Accounting Hub do not have these links.

To support the accounting reversal of transactions accounted prior to the uptake of the Financial Services Accounting Hub, users can use specifically designed upgrade features. In the upgrade case, Financial Services Accounting Hub uses amounts and Accounting Flexfields stored in the transaction objects instead of deriving them from prior subledger journal entries. There are specific accounting attributes defined to deal with the accounting reversals upgrade case.

Assumptions

The accounting reversal upgrade case feature is based on the following assumptions:

- Implementers can create reversal distributions that have access to the entered amounts, ledger amounts, and Accounting Flexfields for the subledger journal entry lines that are subject to the accounting reversal.

- Implementers can derive transaction object lines for the reversed distributions that contribute to no more than two subledger journal entry lines, one debit and one credit.

  Note: A transaction amount such as an invoice line can contribute to any number of journal entry lines. Typically, however, transaction amounts contribute to no more than two journal entry lines; for example, a liability line and an expense line.

  Note: The restriction for the upgrade case dictates that the Financial Services Accounting Hub can only deal with cases where a transaction amount contributes to two journal entry lines.

Transaction Objects

For the accounting reversal upgrade case, the line level transaction objects must contain the following:

- The value U for the Accounting Reversal Option accounting attribute

- The following line accounting attributes that are defined to specifically deal with the upgrade case:
  - Reversed upgrade debit account
  - Reversed upgrade credit account

- The standard sources mapped to the following accounting attributes:
  - Entered amount
  - Ledger amount
  - Rate information
  - Statistical amount
  - Third party identifiers

- Distribution identifiers

> **Note:** In the upgrade case, it is not necessary to populate the transaction object line columns: Reversed Distribution Link Type and Reversed Distribution Identifier.

Instead of searching for distribution links, the Financial Services Accounting Hub creates accounting reversal lines based on the line transaction objects.

If a transaction object line populates both the accounting reversal upgrade debit and credit account accounting attributes, the Financial Services Accounting Hub creates two subledger journal entry lines for the transaction object line. If the transaction object line populates only one of the accounting reversal upgrade debit or credit account accounting attributes, one subledger journal entry line is created by the Financial Services Accounting Hub. In effect, the transaction object passes subledger journal entry lines directly to the Financial Services Accounting Hub.

Note that subledger journal entry lines created by the upgrade case do not have subledger journal entry descriptions or accounting classes.

Accounting Reversal Upgrade Case Example

The invoice shown below is the same as that used in previous examples. However, in this example, it has been accounted prior to the uptake of the Financial Services Accounting Hub.

Upgrade Case: Invoice Validated

The table below lists the details of the invoice header.

Invoice Header

Supplier Name/Number	Invoice Type	Invoice Number
ABC Ltd./XYZ	Invoice	A-1234

The table below lists the details of the invoice distribution.

Invoice Distribution (Validated)

Invoice Distribution Identifier	Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
1234	1	Item	10-Jan-2006	1000	Standard Item
1235	2	Tax	10-Jan-2006	200	VAT @20%

Since it was accounted prior to the upgrade, no transaction object and no distribution links exist. The table below lists the details of the subledger journal entry.

Invoice Adjusted Subledger Journal Entry Lines

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
6000	1	101	Liability	01.4300.000		1200	Lines 2 + 3
6000	2	101	Expense	01.6000.001	1000		Line 2
6000	3	101	Tax	01.4372.000	200		Line 3

Assume that the user now upgrades to the Financial Services Accounting Hub. All subsequent accounting is created using the Financial Services Accounting Hub.

Upgrade Case: Invoice Canceled

On 20-Feb-2006, the invoice is canceled. The table below shows the invoice header for this cancellation.

Invoice Header

Supplier Name/Number	Invoice Type	Invoice Number
ABC Ltd./XYZ	Invoice	A-1234

The table below shows the invoice distribution for the cancellation.

Invoice Distribution (Canceled)

Invoice Distribution Identifier	Invoice Distribution Line Number	Invoice Line Type	GL Date	Amount	Description
1234	1	Item	10-Jan-2006	1000	Standard Item
1235	2	Tax	10-Jan-2006	200	VAT @20%
1400	3	Item	20-Feb-2006	-1000	Cancels line 1
1401	4	Tax	20-Jan-2006	-200	Cancels line 2

The transaction object can take one of two alternative formats depending on whether the debit and credit amounts are supplied on separate lines or merged into a single line. You can decide based on your own criteria whether to merge the debit and credit lines.

The table below lists the transaction object line details for the first format.

Upgrade Case: Invoice Canceled Transaction Objects Lines (Alternative 1 - Subledger Journal Entry Lines are merged)

Invoice Distr. Line Number	Accounting Reversal Option	Invoice Distribution Identifier	Invoice Entered Amount	First Reversed Distribution Identifier	Reversed Upgrade Debit Account	Reversed Upgrade Credit Account
3	U	1400	-1000	Not required for upgrade	01.6000.001	01.4300.000
4	U	1401	-200	Not required for upgrade	01.4372.000	01.4300.000

Alternatively, you need not merge the subledger journal entry lines. The table below lists the transaction object line details for the second format.

Upgrade Case: Invoice Canceled Transaction Objects Lines (Alternative 2 - Subledger Journal Entry Lines not merged)

Invoice Distr. Line Number	Accounting Reversal Option	Invoice Distribution Identifier	Invoice Entered Amount	First Reversed Distribution Identifier	Reversed Upgrade Debit Account	Reversed Upgrade Credit Account
3	U	1400	-1000	Not required for upgrade	01.6000.001	
4	U	1401	-200	Not required for upgrade	01.4372.000	
3	U	1400	-1000	Not required for upgrade		01.4300.000
4	U	1401	-200	Not required for upgrade		01.4300.000

Depending on the ledger level user preference, the Financial Services Accounting Hub creates subledger journal entry lines that reverse the side or sign of the reversed subledger journal entry lines.

Using the new distribution link identifiers, the Financial Services Accounting Hub creates distribution links that tie the Invoice Canceled invoice distributions to the Invoice Canceled subledger journal entry lines.

Assuming that the user has chosen to reverse the sign, rather than the side, the table below lists the details of the Invoice Canceled subledger journal entry. Since this is an upgrade case, there are no accounting classes on the subledger journal entry lines.

Upgrade Case: Invoice Canceled Subledger Journal Entry Lines

Entry Header Identifier	Entry Line Number	Ledger ID	Accounting Class	Account	Debit	Credit	Comment
20000	1	101		01.4300.000		-1200	
20000	2	101		01.6000.001	-1000		
20000	3	101		01.4372.000	-200		

The corresponding distribution link table lines are shown below.

Upgrade Case: Invoice Canceled Financial Services Accounting Hub Distribution Link Table Lines

Entry Header Identifier	Entry Line Number	Ledger ID	First Distribution Identifier	Entered Amount	Ledger Amount	Accounting Reversal Option
20000	1	101	1400	-1000	-1000	U
20000	1	101	1401	-200	-200	U
20000	2	101	1400	-1000	-1000	U
20000	3	101	1401	-200	-200	U

Transaction Objects Example for Payables

To illustrate this example, consider the following invoice:

Invoice Number: A615243

Invoice Date: January 15th

Invoice Due Date: January 31st

The table below provides details of this invoice.

Invoice Details

Line No.	Description	Amount
1	Big Machine	200.00
2	Installation	100.00
3	VAT @ 10%	30.00
	Invoice Total	330.00

Assume that the invoice is in the ledger currency and that the VAT on Installation (line 2) is 50% recoverable.

The transaction object lines for this invoice are seen in the three tables shown below. Note that all three tables are components of the same transaction object lines.

> **Note:** The sources shown in these tables are just a subset of the sources that should be made available.

Invoice Validated Related Sources

Transaction Objects Line No.	Invoice Line No.	Payment Line Type	Invoice Line Type	Entered Amount
1	1	Item		200.00
2	2	Item		100.00
3	3	Recoverable tax	Item	20.00
4	3	Recoverable tax	Item	5.00
5	3	Non Recoverable tax	Item	5.00

Tax Distribution Related Sources

Transaction Objects Line No.	Invoice Line No.	Payment Line Type	Tax Code Account	Tax Name	Recoverable Option	Tax Rate
1	1	Item				
2	2	Item				
3	3	Recoverable tax	01.4700.000	STD Domestic	Y	10.00
4	3	Recoverable tax	01.4700.000	STD Domestic	Y	10.00
5	3	Non Recoverable tax	01.4730.000	STD Domestic	N	10.00

Item Distribution Related Sources

Transaction Objects Line No.	Invoice Line No.	Payment Line Type	Distribution Account	Track as Asset Option	Description
1	1	Item	01.6100.210	Y	Big machine
2	2	Item	01.6000.210	N	Installation
3	3	Recoverable tax	01.6100.210	Y	Big machine
4	3	Recoverable tax	01.6000.210	N	Installation
5	3	Non Recoverable tax	01.6000.210	N	Installation

The above three tables demonstrate the following aspects of transaction object guidelines.

- The transaction object lines are at the lowest level of detail.

  For example, the VAT is allocated to each of the item distributions.

- The transaction object lines contain values for sources drawn from the transaction distribution or allocation and related transaction or distributions.

  For example, the tax distributions contain source values from the item distribution to which they are related.

The following are some examples of requirements that could be met using the above sources:

- Nonrecoverable tax account should be the same as the item expense account. (The account derivation rule uses line type Non Recoverable Tax and item distribution account.)

  The AMB supports two alternative ways of achieving this:

  1. Users can create a single subledger journal entry line which includes the item and related Non Recoverable Tax amount. (The journal line type condition includes line types Item and Non Recoverable Tax.)

  2. Users can create two subledger journal entry lines (Item and Non Recoverable Tax) that use the same account derivation rule.

     See: Account Derivation Rules, *Oracle Subledger Accounting Implementation Guide*

- Non Recoverable Tax account should use an account associated with the tax code. (The account derivation rule uses line type Non Recoverable Tax and tax code account.)

- Liability account should be different for asset and expense items. (The account derivation rule uses line types Item, Non Recoverable Tax, and the Track as Asset flag.) To meet this requirement, the invoice distribution allocation Track as Asset option must be available for both the Invoice Validated and Payment Issued transaction objects.

The linking of related distributions extends across documents. Assuming that the invoice is paid taking a 10% discount, the transaction object lines are provided in the four tables shown below. Note that all four tables are components of the same transaction object lines.

Payment Issued Related Sources

Transaction Objects Line No.	Invoice Line No.	Payment Line Type	Invoice Line Type	Entered Amount (EUR)
1	1	Payment	Item	200.00
2	1	Discount	Item	-20.00
3	2	Payment	Item	100.00
4	2	Discount	Item	-10.00
5	3	Payment	Recoverable tax	20.00
6	3	Payment	Recoverable tax	5.00
7	3	Payment	Non Recoverable tax	5.00
8	3	Discount	Recoverable tax	-2.00
9	3	Discount	Recoverable tax	-0.50
10	3	Discount	Non Recoverable tax	-0.50

Invoice Related Sources

Ext. Line No.	Inv. Line No.	Payment Line Type	Invoice Line Type	Invoice Type	Invoice Number	Invoice Date
1	1	Payment	Item	Standard	A615243	15-Jan-06
2	1	Discount	Item	Standard	A615243	15-Jan-06
3	2	Payment	Item	Standard	A615243	15-Jan-06
4	2	Discount	Item	Standard	A615243	15-Jan-06

Ext. Line No.	Inv. Line No.	Payment Line Type	Invoice Line Type	Invoice Type	Invoice Number	Invoice Date
5	3	Payment	Recoverable tax	Standard	A615243	15-Jan-06
6	3	Payment	Recoverable tax	Standard	A615243	15-Jan-06
7	3	Payment	Non Recoverable tax	Standard	A615243	15-Jan-06
8	3	Discount	Recoverable tax	Standard	A615243	15-Jan-06
9	3	Discount	Recoverable tax	Standard	A615243	15-Jan-06
10	3	Discount	Non Recoverable tax	Standard	A615243	15-Jan-06

Item Distribution Related Sources

Ext. Line No.	Inv. Line No.	Payment Line Type	Invoice Line Type	Distribution Account	Track as Asset Indicator	Description
1	1	Payment	Item	01.6100.210	Y	Big machine
2	1	Discount	Item	01.6100.210	Y	Big machine
3	2	Payment	Item	01.6000.210	N	Installation
4	2	Discount	Item	01.6000.210	N	Installation
5	3	Payment	Recoverable tax	01.6100.210	Y	Big machine

Ext. Line No.	Inv. Line No.	Payment Line Type	Invoice Line Type	Distribution Account	Track as Asset Indicator	Description
6	3	Payment	Recoverable tax	01.6000.210	N	Installation
7	3	Payment	Non Recoverable tax	01.6000.210	N	Installation
8	3	Discount	Recoverable tax	01.6100.210	Y	Big machine
9	3	Discount	Recoverable tax	01.6000.210	N	Installation
10	3	Discount	Non Recoverable tax	01.6000.210	N	Installation

Tax Distribution Related Sources

Ext. Line No.	Inv. Line No.	Payment Line Type	Invoice Line Type	Tax Account	Tax Name	Recoverable Indicator	Tax Rate
1	1	Payment	Item				
2	1	Discount	Item				
3	2	Payment	Item				
4	2	Discount	Item				
5	3	Payment	Recoverable tax				
6	3	Payment	Recoverable tax	01.4700.000	STD Domestic	Y	10.00

Ext. Line No.	Inv. Line No.	Payment Line Type	Invoice Line Type	Tax Account	Tax Name	Recoverable Indicator	Tax Rate
7	3	Payment	Non Recoverable tax	01.4700.000	STD Domestic	Y	10.00
8	3	Discount	Recoverable tax	01.4730.000	STD Domestic	N	10.00
9	3	Discount	Recoverable tax	01.4700.000	STD Domestic	Y	10.00
10	3	Discount	Non Recoverable tax	01.4700.000	STD Domestic	Y	10.00

Note that the payment discount amount is allocated to each of the invoice distribution or allocations.

The following are examples of requirements that could be met using the above sources.

- Liability account should be different for asset and expense items. (The account derivation rule uses line type Payment, allocation types Item, Non Recoverable Tax, and Track as Asset flag.)
- Discount account for Non Recoverable Tax should be the same as the item expense account. (Account derivation uses line type Discount, allocation type Non Recoverable Tax, and the item distribution account.)

4

Seeding Event Information Using Accounting Methods Builder

Seeding Event Information Using Accounting Methods Builder (AMB) Overview

This chapter describes:

- How the Financial Services Accounting Hub creates subledger journal entries from accounting events using event entities, event classes, and event types
- How to register the Financial Services Accounting Hub application, which is required before using the AMB

Event Setup Steps

Along with the definition of sources, the accounting events setup establishes the framework within which users create and modify their application accounting definitions. The following steps, also shown in the Event Setup Overview diagram below, must be taken to seed event information using the AMB:

- Sources and accounting attributes
  1. Run, create, and assign sources: modify sources.
  2. Assign sources to accounting attributes.

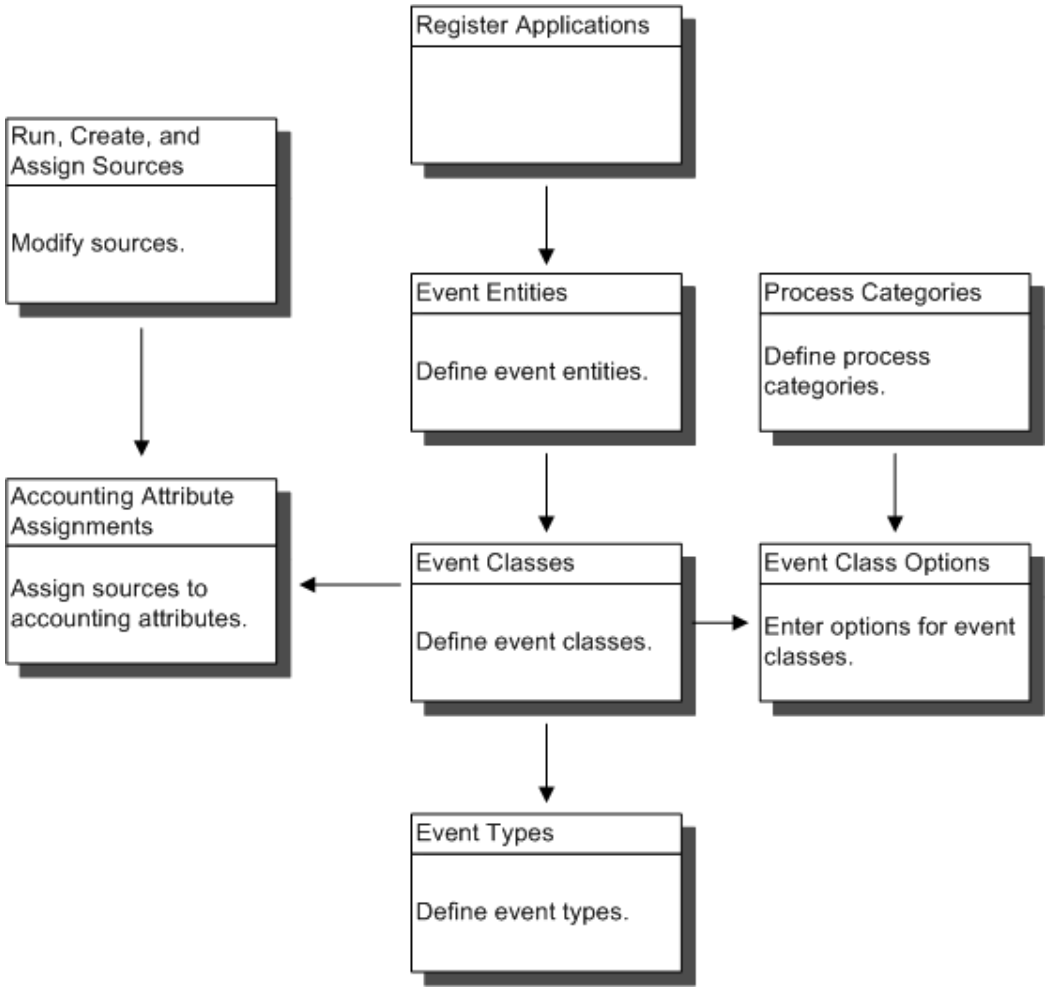
     See: Assigning Sources, page 6-13

- Events
  1. Register applications.

2. Define event entities.
3. Define event classes.
4. Define event types.
5. Define process categories.
6. Enter options for event classes.

These components are fully described in Revise Source Definitions and Assign Accounting Attributes Introduction, page 6-1.

Event Setup Overview



After registering the application, setting up event entities, event classes, and event types are prerequisites. Sources must also be defined for the application. Once these are set up, sources are assigned to event classes. Lastly, Process Categories are defined for applications and event class options are defined for event classes.

Introduction to Event Entities, Event Classes, and Event Types

Accounting events are the basis for creating subledger journal entries. Within the events model, accounting events are categorized by event type, event class, and event entity.

See: Overview of the Financial Services Accounting Hub, page 1-4

At the lowest level, accounting events are categorized by event type. Accounting events with significantly different fiscal or operational implications are classified into different accounting event types, such as Invoice Approval, Invoice Adjustment, and Invoice Cancellation. The accounting program uses event types, along with accounting setups

and application accounting definitions, to determine which subledger journal entries will be created.

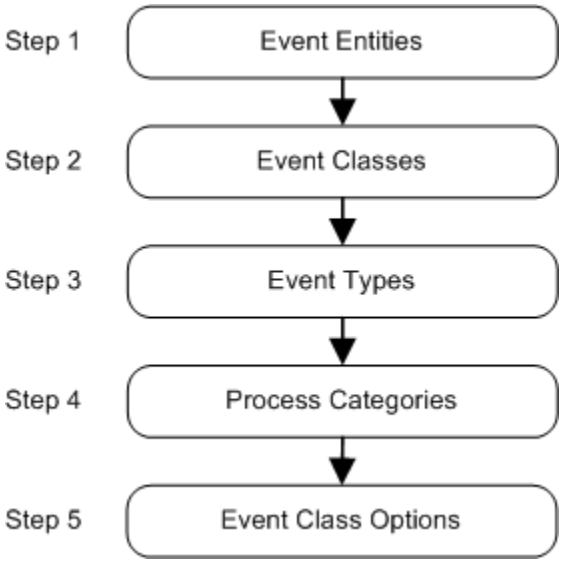
The Financial Services Accounting Hub recognizes that while there are many kinds of accounting event types, there are also common requirements which they share. It is important to balance the need to create unique accounting definitions for each event type with the goal of an efficient implementation. To achieve this balance, the Financial Services Accounting Hub provides event entities and event classes.

An event class groups similar event types together for the purpose of sharing sources and accounting definitions. This reduces the setup time as it eliminates the need for separate accounting definitions for each event type. Event entities group related event classes. Event classes within an event entity usually share the same entity table in the transaction data model. For example, in Payables, invoices and prepayments can be grouped under the same event entity as both of them are stored in the table AP_INVOICES_ALL.

Event entities, classes, and types are specific to an application. Implementers are responsible for identifying and seeding all accounting event types, classes, and entities for their applications in the AMB.

For each accounting event, the Load Transaction Objects program provides the accounting program with source values which, in conjunction with the application accounting definitions, are used to create subledger journal entries.

As this chapter is restricted to the seeding of event information in the AMB, the following steps are a subset of the procedures shown in the Event Setup Overview, page 4-3 diagram. Each of these steps corresponds to a particular window in the AMB and is described in detail in a subsequent section of this chapter.



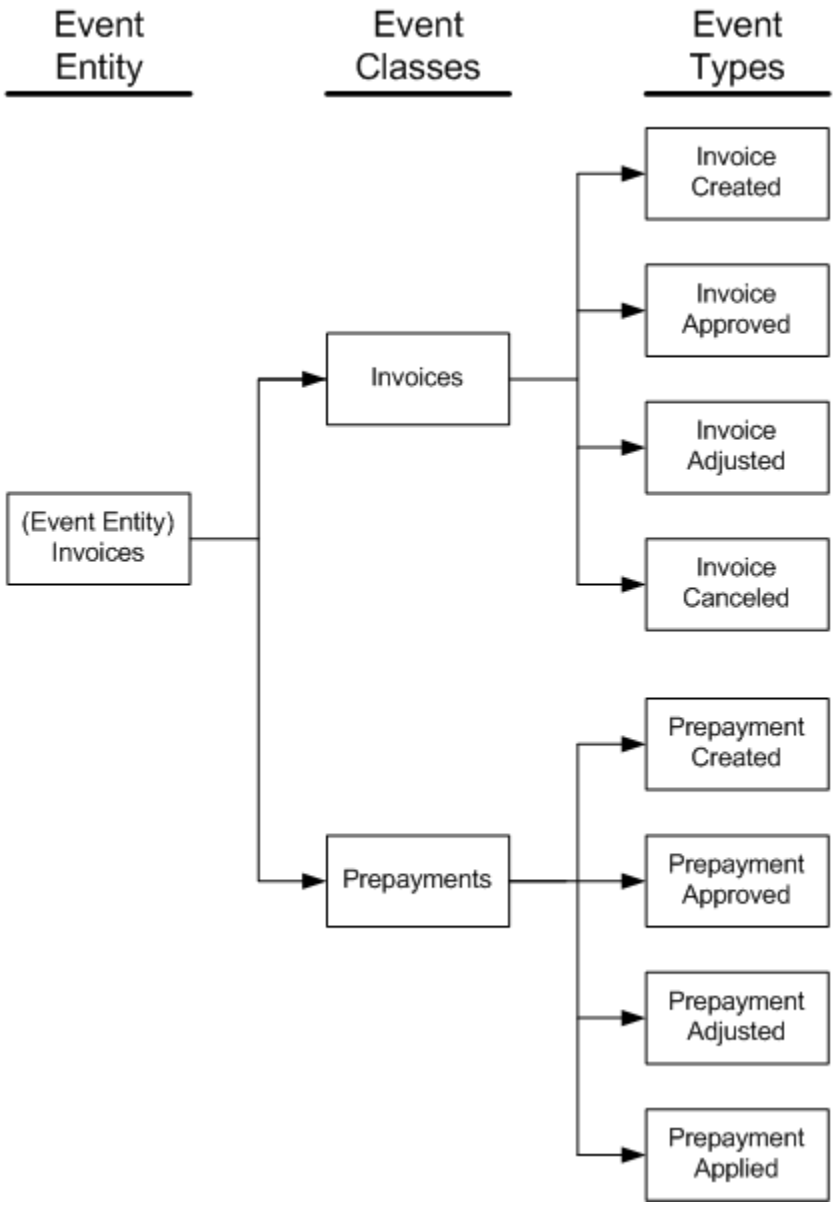
The above diagram illustrates the component definitions in the following order:

1. Step 1: Event Entities

2. Step 2: Event Classes
3. Step 3: Event Types
4. Step 4: Process Categories
5. Step 5: Event Class Options

Example of Entities, Classes, and Types

The diagram below provides an example of the relationship between event entities, event classes, and event types.



As shown in the above diagram, a single event entity can be associated with multiple event classes. The event entity Invoices is associated with the Invoices and Prepayments event classes.

Each of these event classes in turn, can be related to several event types. The Invoices event class is associated with the following event types:

- Invoice Created
- Invoice Validated
- Invoice Adjusted

- Invoice Canceled

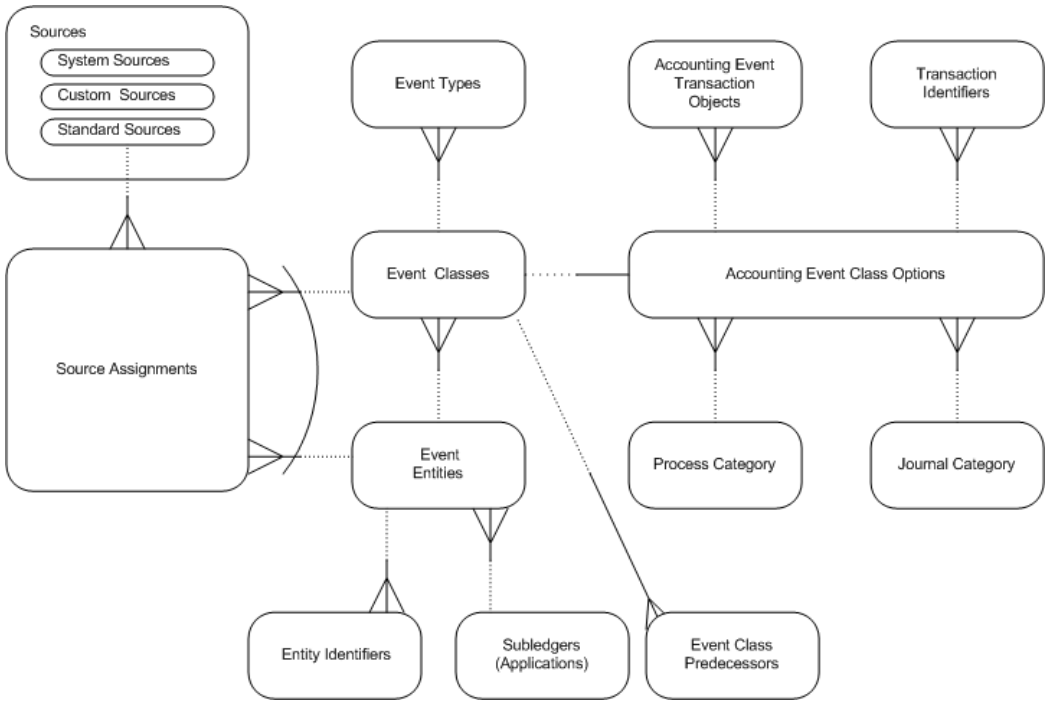
The Prepayment event class is associated with the following event types:

- Prepayment Created
- Prepayment Approved
- Prepayment Adjusted
- Prepayment Applied

Events Data Model

The entity relationship diagram below describes the entities related to the setup of accounting events and is described in the subsequent sections.

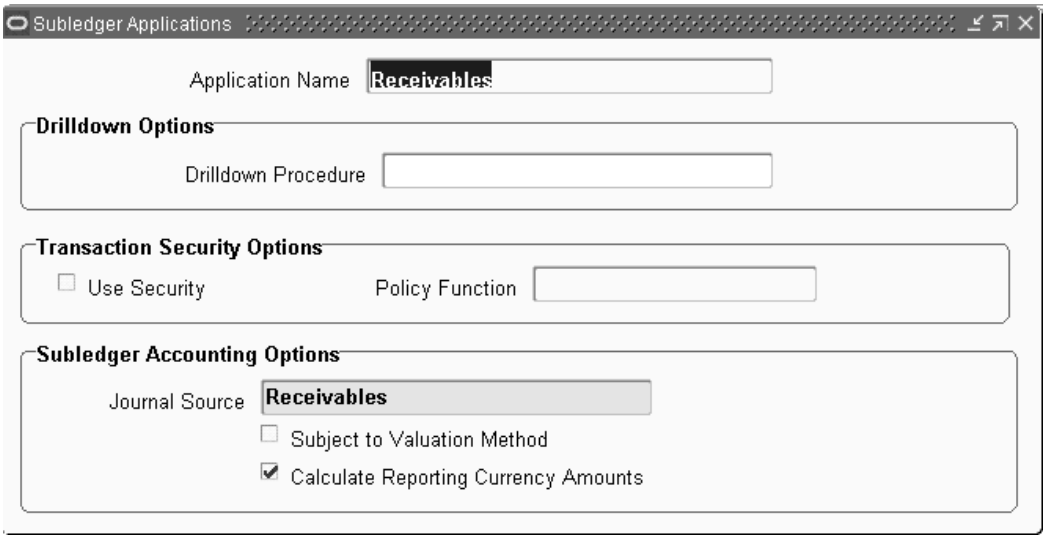
Events Data Model



Registering Subledger Applications

To use the AMB, register your application with the Financial Services Accounting Hub. Specify options related to event security, General Ledger, and subledger accounting options along with the names of event processing procedures.

To Register Subledger Applications



Selected Fields in the Subledger Applications Window

Field	Description
Application Name	List of values includes all existing Oracle E-Business suite applications. To display external applications, register them using the application's developer responsibility. If an application is registered, query the application.
Drilldown Procedure	Stored procedure that the Financial Services Accounting Hub uses to drill down from a journal entry to the corresponding subledger transaction See: Drill-down API Details, page 13-2
Use Security	If not selected, then there is no event security for this application and a session has access to all events in the application regardless of the transaction's security.

Field	Description
Policy Function	If Use Security is selected, enter a stored function that establishes security access for a session. Because this field has no validation, ensure that the value entered corresponds to a valid database function. See: Financial Services Accounting Hub Security Introduction, page 11-1
Journal Source	Identifies the origin of the subledger journal entries
Subject to Valuation Method	Select if the application supports multiple valuations for a single transaction. See: Valuation Methods and the Financial Services Accounting Hub, page 2-24

Field	Description
Calculate Reporting Currency Amounts	Select to calculate accounted amount for any reporting currency. If selected, the Create Accounting program calculates accounted amounts and creates journals for all enabled subledger level reporting currencies associated with the primary ledgers. If not selected, the Create Accounting program does not calculate accounted amount for any reporting currencies, but it does create journals for the subledger level reporting currencies if reporting currency information is provided in the transaction objects. If the Create Accounting program fails to create journal entries for any of the subledger level reporting currencies, no journal entries are created for the primary and secondary ledgers and for other subledger level reporting currencies. This option does not apply to secondary ledgers. The Financial Services Accounting Hub calculates accounted amount for secondary ledgers regardless of the value of the Calculate Reporting Currency Amounts option. **Note:** Select this check box for products that support Multi-reporting Currencies in release 11.5 except for Oracle Assets and Oracle Projects. **Note:** This option does not correlate to the Calculate Accounted Amounts option in the Accounting Event Class Options window. The Calculate Accounted Amounts option applies to currency conversions from transaction currency to the functional currency of the ledger that the event is associated with. The Calculate Reporting Currency Amounts option applies to currency conversions to

Field	Description
	subledger level reporting currency.
Third Party Type	Select Supplier, Customer, Any, or None.

Drilldown API Details

See: Drill-down API Details, page 13-2

Event Entities

Use accounting event entities to treat events for a single transaction model in a similar manner. Event classes within an event entity usually share the same entity table in the transaction data model and can be uniquely identified by the same primary or surrogate key.

Purpose

Use event entities to tie subledger transaction entities with their accounting representation. The Financial Services Accounting Hub uses event entities to maintain the system transaction identifiers that uniquely identify the subledger transactions associated with the accounting events.

Example: Oracle Payables invoices and prepayments are stored in the same entity table in the data model and are identified by the same primary key. Therefore, they can be grouped under the same event entity.

Characteristics

Event entities have the following characteristics:

- They usually correspond to a database object.

  They can also logically correspond to a single transaction, which is used as a basis for several related accounting events. Examples of such transactions include Payables or Receivables invoices.

- Event entities differ by application; implementers create their event entities for a particular application.

Consider the following issues:

- Whether different types of transaction data are maintained in the same table

  Such transactions should typically be part of the same event entity. For example, in Payables, invoices, prepayments, and credit memos are stored in the same table. Therefore, it is logical to have an event entity to cover these accounting event classes. The table below lists examples of event entities for Payables and their associated entity tables.

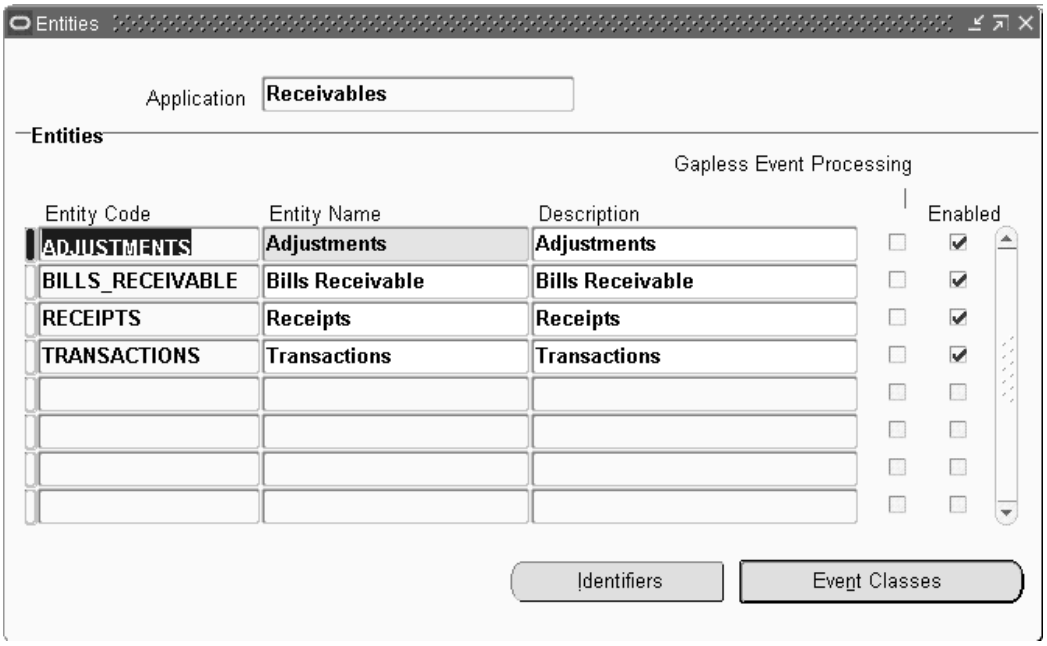
Event Entity	Entity Table Name
Invoices	AP_INVOICES_ALL
Payments	AP_CHECKS_ALL

- Whether there is a requirement to provide separate sequences for the different kinds of subledger journal entries used to account for events

Defining Accounting Event Entities

Use the Entities window to set up event entities and system transaction identifiers for the application. This window is only available to implementers. Users cannot create or update event entities.

To Define Accounting Event Entities



Selected Fields and Buttons in the Entities Window

Field or Button	Description
Entity Code	Implementer's key used by programs and routines to refer to the entity. Codes are used because event entity names are translated.
Description	If the event entity represents a specific transaction, then include the transaction name. Otherwise, include a description of the types of transactions represented by the event entity.
Gapless Event Processing	Enables gapless event processing. If selected, all the events associated with the entity are flagged as On Hold if there is a gap in the event numbers.
Enabled	If selected, makes this definition available for use

Field or Button	Description
Identifiers	Opens the System Transaction Identifiers window
Event Classes	Opens the Event Classes and Types window

System Transaction Identifiers Window

You can assign up to five attributes to uniquely identify instances of an event entity. These attributes constitute the primary or surrogate key for the entity tables in the transaction data model where the subledger transaction associated with the event entity are stored. These identifiers link accounting events with their underlying transactions in the transaction data model. Define the link by mapping columns from the entity table to system transaction identifiers for the event entity in the System Transaction Identifiers window.

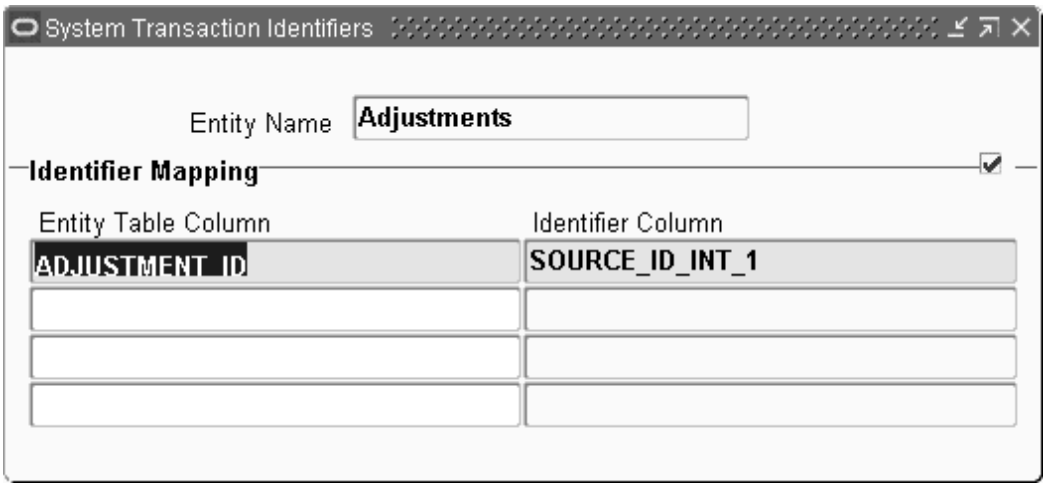
> **Note:** The System Transaction Identifiers window provides only the definition of the link and not the actual link. The latter is stored in the XLA_TRANSACTION_ENTITIES table.

As part of registering event entities in the AMB, system transaction identifiers are assigned to entity table columns in the transaction data model.

See: System Transaction Identifiers, page 2-8

In the System Transaction Identifiers window, enter the columns that constitute the transaction header table's primary or surrogate key and assign them to system transaction identifiers.

To Define a Link Between an Accounting Event and Its Associated Transaction



Selected Fields in the System Transaction Identifiers Window

Field	Descriptions
Entity Table Column	Stores the column names that constitute the primary or surrogate key of an accounting event's underlying subledger transaction
Identifier Column	Corresponding identifier column name; stores the column names from the transaction entities table that the Entity Table column is mapped to List of values for this field includes 15 identifiers; five of these columns are numeric, five are alphanumeric, and five are dates. When events are captured, the event identifiers must be passed to the event capture APIs in the corresponding parameters. For example, if TRANSACTION_ID is mapped to SOURCE_ID_INT_1 on this window, then the value of TRANSACTION_ID must be passed to the accounting event APIs in the SOURCE ID_INT_1 parameter. See: Introduction to Capturing Events, page 8-1

Event Classes

Accounting event classes group accounting event types into distinct, user-oriented groups. They typically represent the actions possible on a particular transaction or transaction type.

For example, the Payables Invoice transaction model stores two different types of user transactions: invoices and prepayments. Receivables stores invoices, deposits, guarantees, and bill receivables in the same transaction table, but to the user, they are distinct types of transactions. Each of these transaction types lends itself to being an event class. Typically, there are different event classes for each category of real world transaction or transaction type.

Examples of real world transactions that normally correspond to accounting event classes are invoices, payments, purchase orders, receipts, and lease contracts. Each of these are associated with different kinds of transactions and account for business events with different operational and fiscal significance.

Examples of different types of transactions that result in distinct event classes are asset additions, depreciation, and retirements. While these operations are based on the same object (the underlying asset), they correspond to separate transactions that require different accounting treatment. Each of these operations results in an association with different event classes.

Accounting event classes are visible to users.

Purpose and Characteristics

Accounting definitions usually differ by accounting event class. Use event classes to group similar event types for sharing accounting definitions. In other words, assign accounting definitions to be used by all accounting event types in the class at the event class level.

In addition, assign sources and journal line types assigned to event classes. These sources and line types are available for creating accounting definitions for any event type in the event class. Also, default assignments for accounting attributes are made at the event class level.

Event classes enable users to maintain accounting and reporting sequences at the subledger journal entry level.

See:

- Assigning Sources, page 6-13
- Journal Line Types, *Oracle Subledger Accounting Implementation Guide*
- To Assign Accounting Attributes in the Accounting Methods Builder, page 6-15

Not all sources are relevant to all subledger journal entries. You can limit which event

classes can use a given source to create subledger journal entries.

The following example explains why sources are restricted. Consider a source such as Receipt Date. An implementer may want to make this source available to the Receivables implementers for creating subledger journal entries. However, this source is irrelevant to users trying to create a subledger journal entry for the completion of Payables invoices. Restricting the Receipt Date to where it can be used avoids confusion.

Consider the following issues when defining event classes:

- What users regard as distinct, user-oriented transaction groups
- Whether sources are shared
- Items such as descriptions and accounts that can be common to related transactions or transactions
- Whether all related event types belong to a common group

  For example, credit memos can be considered a separate event class from transactions if there is an event type that is relevant only to credit memos.

  If credit memos can be applied only to invoices, consider creating an event class for credit memos.
- Whether there is a requirement to provide separate sequences for the different kinds of subledger journal entries used to account for events

Defining Event Classes

Use the Event Class and Types window to define event classes for the application after creating event entities. This window is accessed from the Entities window. All event classes created relate to the same associated event entity. Users cannot create or update event classes.

Define an event class for:

- Each set of accounting events that pertain to the same kind of transaction
- Sources to be shared

See: To Define Event Classes and Types, page 4-19

Update Subledger Accounting Options Setup Concurrent Program

This program performs the default Financial Services Accounting Hub options setup on demand. It needs to be executed in the following circumstances:

- When event classes are added or modified after ledger accounting setup has been completed, users must run this program to include the new event classes in the

accounting setup.

- In a running install where ledgers are implemented, implementers can decide to uptake the Financial Services Accounting Hub for a custom subledger application. Once the custom subledger application is set up in the AMB, users need to run this program to set the defaults for the custom subledger application and all the implemented ledgers.

Event Types

Accounting event types represent the business operations that can be performed on the event class. For example, the Payables event class Invoices is subject to three types of business operations, which are also event types: Invoice Validated, Invoice Adjusted, and Invoice Canceled.

An accounting event always has an associated event type. For products that are based on transactions, event types correspond to the different operations that can be performed on the transactions. For products that are nontransaction based, an event type corresponds to a particular kind of transaction.

Event types that pertain to the same kind of transaction and are subject to a similar accounting treatment should be associated with the same event class.

Define a separate event type for each kind of accounting event that is likely to require a different subledger accounting treatment.

Purpose

Accounting event types, used to categorize accounting events, provide the most granular level of detail for creating accounting definitions. Accounting events with significantly different fiscal or operational implications should be classified into different event types. Event types do not necessarily correspond to system operations. A business perspective yields the best definition for an accounting event type.

Characteristics

By virtue of providing the most granular level of detail, accounting event types have the following characteristics:

- Enable accounting definitions based on the event type

  This is useful when an environment has detailed and varying accounting requirements.

  In addition, event types provide an accurate name and description of the business event which gave rise to the subledger journal entry.

  For example, using event types enables users to distinguish between subledger journal entries that arise from accounting events invoice approvals, invoice

adjustments, and invoice cancellations. As another example, users can create different descriptions for canceling and applying receipts. Creating separate event types for these two operations makes this possible.

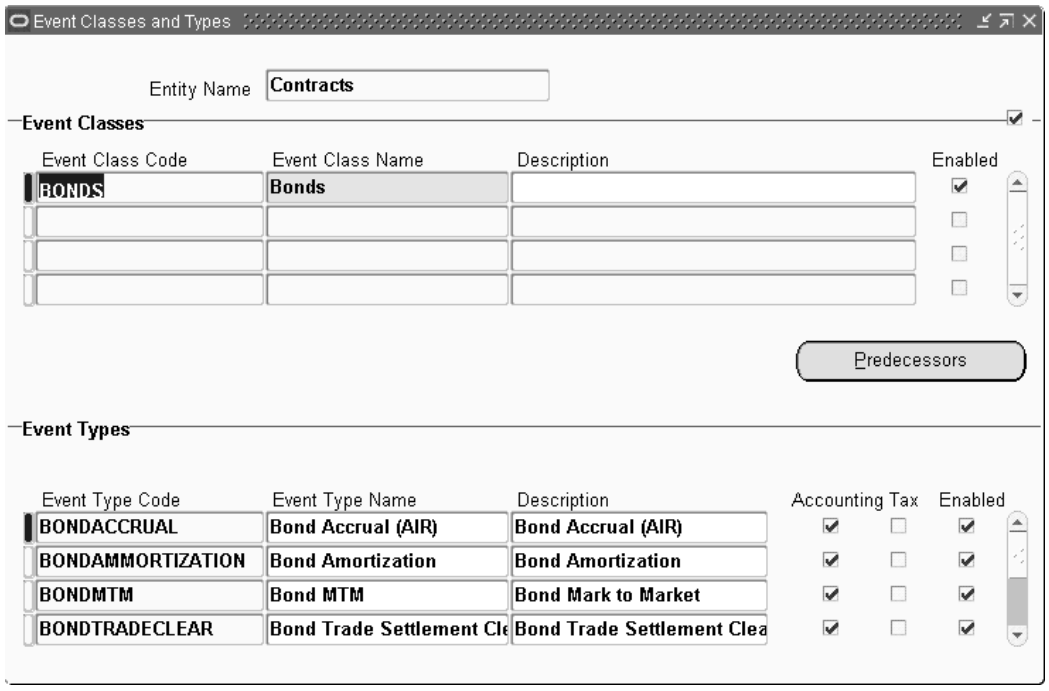
- Provide an audit trail of business events on a transaction
- Visible to end users

Consider the following when defining accounting event types:

- Whether a particular transaction or occurrence creates accounting events that require a distinct accounting treatment
- Whether users need a separate accounting event type to be able to understand the relationship between the actions they perform in the subledger products and the accounting for the associated accounting event

  For example, since there is a separate accounting event type called Invoice Adjusted, users understand that this event type is used to account for invoice adjustments.

To Define Event Classes and Types



Selected Fields and Buttons in the Event Classes and Types Window

Field or Button	Description
Event Class Code	Developer key used by programs and routines to refer to this event class. Codes are used because event class names are translated.
Event Class Name	Create user-oriented names because they appear in several AMB windows.
Enabled	If selected, makes this definition available for use
Event Type Code	Developer key used by programs and routines to refer to this event type. Codes are used because event class names are translated.
Event Type Name	Application accounting definitions are based on event types. It is critical that all event type names have a user orientation and are consistent with the user's experience in the subledger applications. As an example of the kind of naming to be employed, event types for invoices should not be called Invoice Inserted, Invoice Updated, and Invoice Deleted. Use names like Invoices Created, Invoice Adjusted, and Invoice Canceled. Define an event type in the system before events can be captured for that event type. Define event type names and descriptions in the past tense because the event is captured only if the transaction has completed successfully.
Accounting	Indicates whether the event type is used by the Financial Services Accounting Hub
Tax	Indicates whether the event type is used for tax and sequencing
Enabled	If selected, makes definition available for use

Field or Button	Description
Predecessors	Opens the Event Class Predecessors window

Event Class Predecessors

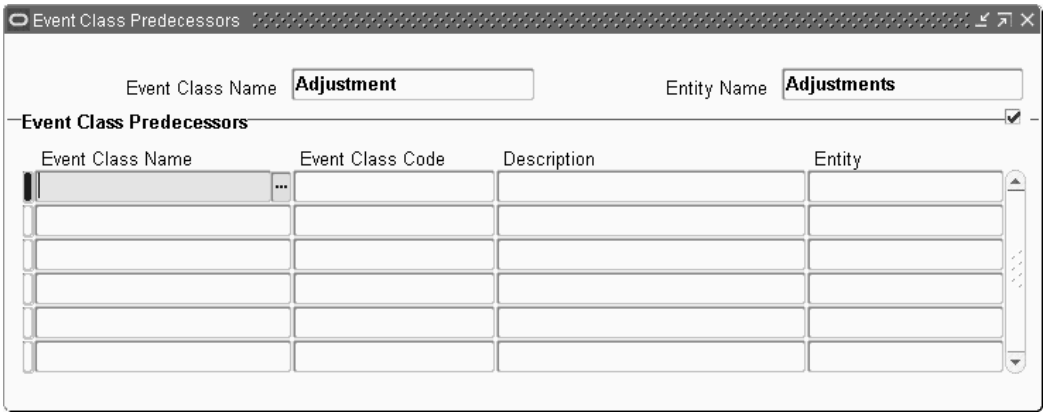
Event class predecessors ensure that if there is an accounting event that is dependent upon the accounting of a prior event, the accounting of the dependent event is not delayed if the prior event is not yet accounted. For example, without event class predecessors, an AR Receipt that is dependent on an AR Invoice cannot be accounted if the invoice is not accounted. With event class predecessors, the Create Accounting program processes the AR Invoice before processing the AR Receipt thereby minimizing delays in accounting due to the dependencies inherent in business flow processing.

Predecessors can exist across entities and are defined at the application level. They must be in the same application because the Create Accounting program runs for only one application at a time. If an event is dependent on the accounting of another event that is in a different application, accounting must still be run separately for that application before the dependent event can be accounted.

Defining Event Class Predecessors

The Event Class Predecessors window establishes an order in which the Create Accounting program processes events selected for accounting. This order of events ensures that the accounting of a dependent event is not delayed because it depends on an unaccounted prior event. This window is accessed from the Event Classes and Types window.

To Define Event Class Predecessors



Selected Fields in the Event Class Predecessors Window

Field	Description
Event Class Name Entity Name	Default from the Event Class and Types window
Event Class Name	Event class that the Create Accounting program must process before accounting the event class that is displayed in the header region of this window List of values has the following attributes: • Displays the event classes for the same application of the selected event class indicated in the header region • Filters out the currently selected event class to prevent selecting an event class as a predecessor of itself • Filters out event classes that are not enabled • Displays the event class name, event class code, application, and entity The Event Class Code, Description, and Entity fields are automatically populated with related information.

Process Categories

Process categories are collections of one or more logically related event classes. They restrict the events selected for accounting when users submit the Accounting Program. Events can be restricted from separate processing because of possible dependencies between transactions of different event classes. Such dependencies mean that accounting events of one class should not be accounted independently from accounting events of another class.

By grouping event classes so that they can be processed in a single submission of the Accounting Program, process categories enable users to create and transfer accounting for all the accounting events in a business flow.

For example, in many situations, accounting events for the Payment Issue event class should not be accounted independently from accounting events for the Payment Clear event class. These should be processed by a single submission of the Accounting Program. To accomplish this, group these two event classes into a single process category.

Seed a process category for each set of event classes that composes a business flow and that you want to create accounting for in a single run of the Accounting Program.

When submitting the Accounting Program, users can choose the process category that they want to create accounting for. If no process category is specified, the selection of accounting events to be processed is not restricted by event class.

> **Note:** Assign event classes to process categories in the Accounting Event Class Options window.

See: Accounting Event Class Options, page 5-1

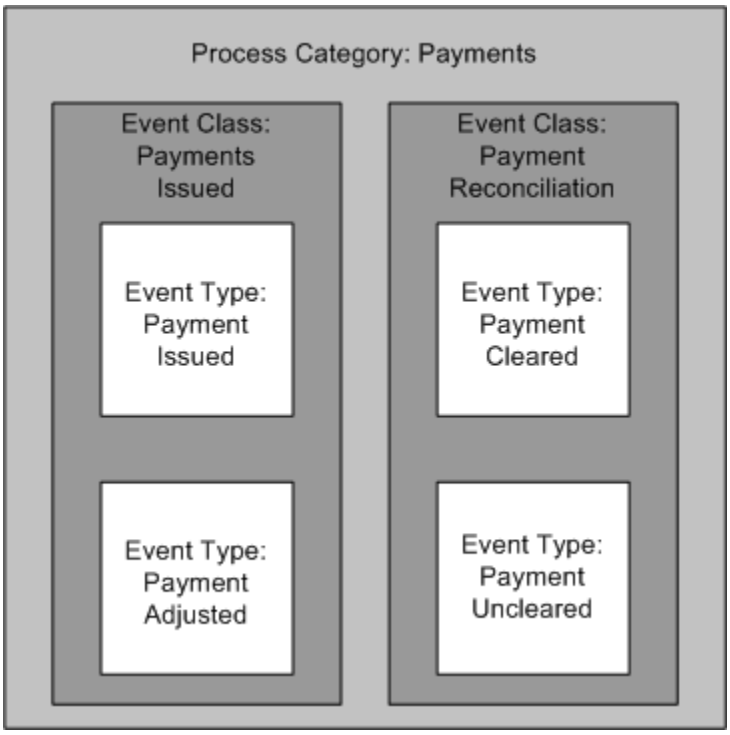
Selecting a process category indicates that all associated event classes and their assigned event types are selected for processing.

Process Categories Example

Consider the example of a cash manager who wants to reconcile General Ledger cash account balances to bank statements received.

To undertake this reconciliation, the first step is to process all transactions pertaining to cash to obtain an accurate report of the organization's cash account balances. The manager must ensure that all entries affecting the cash balance in General Ledger are selected for processing. This includes entries related to payments, their clearing, and unclearing.

A sample process category setup is shown in the figure below and is described in the Process Category: Payments table, page 4-24.

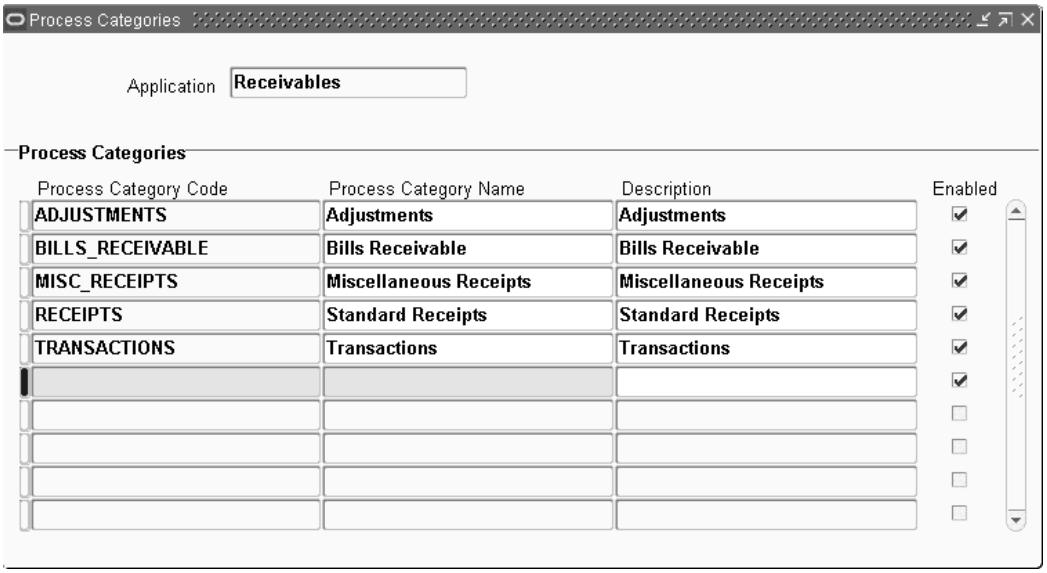


Process Category: Payments

Event Class: Payments Issued	Event Class: Payment Reconciliation
Event Type: Payment Issued	Event Type: Payment Cleared
Event Type: Payment Adjusted	Event Type: Payment Uncleared

In this example, the cash manager can select the process category Payments when the Accounting Program is submitted. As a result, all events associated with these event types are selected for processing by the Accounting Program.

To Define Accounting Process Categories



The table below describes selected fields from the Process Categories window.

Process Categories

Field	Description
Process Category Code	Developer's key used by programs and routines to refer to this category
Process Category Name	Selected by users when submitting the Accounting Program

5

Create and Assign Sources

Create and Assign Sources Overview

After creating transaction objects, assign these objects to event classes when setting up event class options in the Accounting Methods Builder (AMB).

Run the Create and Assign Sources program to automatically generate sources and source assignments based on transaction object and reference object definitions and assignments. These sources are used to create journal entries for transactions in the subledger.

Defining Accounting Event Class Options

In the Accounting Event Class Options window, you can:

- Enter additional information for event classes

  For example, for events belonging to a particular event class, enter supplemental information such as the default journal category, which is used to create journal entries in General Ledger.

- Assign transaction objects to event classes

- Define reference objects

- Define relationships between transaction objects and reference objects

  For example, a Payables implementer can define Payables table AP_PAYMENT_SCHEDULES and Projects table PA_PROJECTS as reference objects assigned to the transaction object AP_INVOICES for an event class Invoice.

- Define relationships between primary and secondary reference objects.

To Set Up Accounting Event Class Options

The following procedure describes selected fields in the Accounting Event Class Options window.

1. The Application field defaults from the application associated with the responsibility.

2. In the Event Class field, select the event class from the list of values.

   The list of values for the Event Class field shows all event classes associated with the entity selected.

   If accounting event class options are defined for the event class, then the options can be queried.

3. In the Process Category field, enter the process category associated with this event class from the list of values.

   The list of values for this field contains all process categories for the application associated with this event class.

   See: Process Categories, page 4-23

4. In the Journal Category field, select a journal category from the list of values.

   For events belonging to this event class, the General Ledger Journal Category field stores the default journal category that is used to create journal entries in General Ledger. The list of values for this field contains all journal categories defined in General Ledger. These categories can be overwritten when users complete the Financial Services Accounting Hub setups in Accounting Setup Manager.

   See: *Oracle Financials Implementation Guide*.

5. In the Transaction View field, enter the view name that can be used to retrieve additional information for an event from the list of values.

   The information acquired from this view is displayed in the Financial Services Accounting Hub reports and inquiries.

   You should provide a transaction view for each event class. This view should include all columns that have been mapped to system transaction identifiers for the event entity as well as those identifiers that may help users to uniquely identify the transaction in the application. The Financial Services Accounting Hub uses these values to select information from the view.

   Many event classes can share a single transaction view. For example, the event class Prepayments can use the same transaction view as the event class Invoices. The transaction view column names can vary from event class to event class. For example, label the INVOICE_NUM column Invoice Number or Prepayment Number depending on the event class.

   In the User Transaction Identifiers tab, you can specify up to ten columns from the transaction view that appear on the Financial Services Accounting Hub inquiry windows and reports. These identifiers uniquely identify a document or transaction in the application.

   See: User Transaction Identifiers, page 2-8

6. In the Balance Types region, select the balance types supported by the event class.

   - Select Budget for each event class that may have budget accounting lines.
   - Select Encumbrance for each event class that may have encumbrance accounting lines.

     See: Budgetary Control Accounting Attributes, page 3-35

7. Optionally, to configure the Financial Services Accounting Hub to calculate accounted amounts for the event class, select the Calculate Accounted Amounts check box in the Currency Conversion Options region.

   The Calculate Accounted Amounts option is for the primary ledger or secondary ledger with valuation method enabled. Subledger accounting calculates the accounted amount for the secondary ledger if the secondary ledger is in a different

currency than the primary ledger and the secondary ledger is not enabled for valuation method.

8. Optionally, to configure the Financial Services Accounting Hub to calculate gain or loss amounts for the event class, select the Calculate Gain or Loss Amounts check box in the Currency Conversion Options region.

   The table below describes the viable combinations for the Calculate Accounted Amounts and Calculate Gain or Loss amounts options.

Calculate Accounted Amounts	Calculate Gain or Loss Amounts	Financial Services Accounting Hub Behavior	Implementers Provide in the Transaction Objects
Yes	Yes	The Financial Services Accounting Hub calculates both accounted and gain or loss amounts.	If the conversion rate type is User, then you should also provide the conversion rate; otherwise, provide the conversion date.
No	Yes	The Financial Services Accounting Hub calculates gain or loss amounts only, including any rounding amounts for the event class.	Unrounded accounted amounts, conversion information: conversion rate type, conversion date, or rate type
No	No	The Financial Services Accounting Hub does not perform any calculations.	Unrounded accounted amounts, conversion information: conversion rate type, conversion date, or rate type Accounted amounts and gain or loss amounts

Note: Conversion information, which includes conversion rate type, conversion date, or conversion rate, is always required for foreign currency.

9. In the Sequence field, enter a Sequence number to specify the column order that appears in the Financial Services Accounting Hub inquiry windows and reports.

   The Sequence Number field automatically generates sequential numbers in ascending order. However, users can change the defaults.

10. In the User Identifier Name field, enter the label to use when the identifiers are displayed on reports and in subledger accounting inquiry windows.

    This field is translated so that user transaction identifiers are always displayed with labels in their preferred language. Users can modify this name to meet their requirements.

11. In the View Column Name field, select the view column to display on accounting reports and inquiry windows.

    You can specify up to five column names that provide additional information about the event.

    The list of values for this field contains all columns from the view entered in the Transaction View field.

    The Financial Services Accounting Hub displays user transaction identifiers on accounting event reports and inquiries by dynamically generating sql statements based on the definitions of both system and user transaction identifiers. Details are obtained as follows:

    - The Transaction View defines the database table or view with the user transaction identifiers.
    - The View Column Names field defines the columns to be selected from the database table or view.
    - The system transaction identifiers define the WHERE clause applied to the database view or table. The value of these unique identifiers are taken from the accounting event data model.

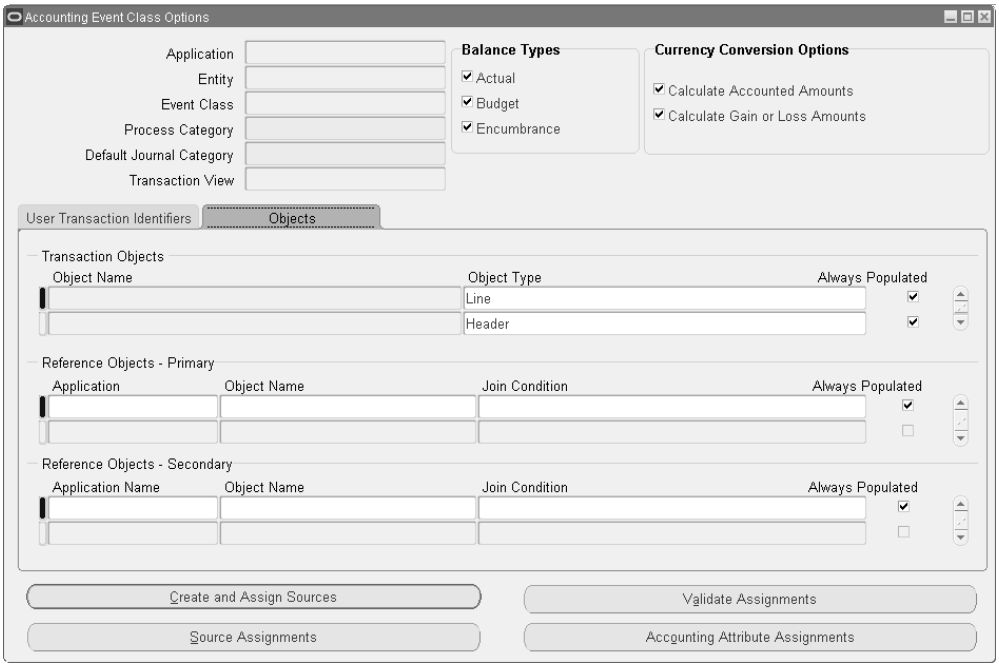
    As an example, the Financial Services Accounting Hub builds the following select statement to retrieve the user transaction identifiers for accounting events for the Invoices and Prepayment event classes:

```
SELECT VENDOR_NAME,
VENDOR_NUMBER,
INVOICE_NUM,
INVOICE_DATE
FROM AP_INVOICES_TRANSACTION_LINK_V
WHERE INVOICE_ID = &SOURCE_ID_INT_1;
```

    You must specify the tables or views from which the accounting transaction object takes source values for a particular event and transaction object line.

See: Introduction to Events and Sources, page 1-3

See: Transaction Objects Introduction, page 3-1



12. In the Object Name field in the Transaction Objects region of the Objects tab, enter the table or view name corresponding to the transaction object.

    Object names must be unique inside an event class.

13. In the Object Type field, select the information provided by the sources included in the transaction object from the list of values.

    The list of values contains the following transaction object types:

    - Transaction Objects Headers: for untranslated header level sources
    - Transaction Objects Headers MLS: for translated header level sources
    - Transaction Objects Lines: for untranslated line level sources
    - Transaction Objects Lines MLS: for translated line level sources

14. For each transaction object type, select the Always Populated check box for at least one transaction object.

    The Always Populated check box is selected by default.

    If Always Populated is selected, the Accounting Program expects that the transaction object specified always returns one or more records for the accounting event. If the rows are not returned, an error message appears and journal entries are

not created.

If Always Populated is not selected, the transaction object does not always need to return records.

For example, since not all Payables invoice lines are matched to purchase orders, a transaction object lines object that is not Always Populated can be used to store all sources that relate to purchase orders. It returns rows only for those invoice lines that are matched to purchase orders.

15. In the Application field of the Reference Objects region, select the application of the primary or secondary reference object.

16. In the Object Name field, enter the object name of the reference object.

    The object must exist in the database and be accessible to users. The object name must be unique within an event class.

17. In the Join Condition field, enter the join condition between the transaction and reference objects.

    Note: Implementers cannot use the outer join operator (+).

18. Select the Always Populated check box if the join condition always returns a record.

    If this check box is not selected, the Financial Services Accounting Hub creates the outer join.

    Note: The Always Populated flag of a transaction object does not alter join conditions of reference objects.

See: Reference Objects, page 3-13

Create and Assign Sources Program

The Create and Assign Sources program derives sources and source assignments given the definitions of transaction objects and reference objects. The tool ensures consistency between transaction and reference objects and source and source assignment definitions. The program validates transaction objects, reference objects, source definitions, and source assignments. Optionally it can also validate accounting attribute assignments.

With the Create and Assign Sources program, you are only responsible for extending source definitions to include optional user attributes. You also need to add accounting attribute assignments.

The Create and Assign Sources program is generated by clicking **Create and Assign Sources** in the Accounting Event Class Options window.

See: Revise Source Definitions and Assign Accounting Attributes, page 6-3

Based on the transaction and reference object definitions and corresponding database data dictionary definitions, the Create and Assign Sources program can be launched in the following modes:

- Create and Validate, page 5-8
- Validate Only, page 5-9

Create and Validate Mode

Select the Create and Validate mode when launching the Create and Assign Sources from the Submit Request window or by selecting the Create and Assign Sources button in the Event Class Options page. When run in the Create and Validate mode, the Create and Assign Sources program performs the following tasks:

- Validation of Transaction and Reference Objects Defined in the AMB, page 5-8
- Creation of Sources, page 5-8
- Assign Sources, page 5-9

Validation of Transaction and Reference Objects Defined in the AMB

The Create and Assign Sources program verifies the following for transactions and reference objects:

- All transaction and reference objects defined for the event class exist in the database data dictionary.
- All transaction objects, based on the transaction object type, contain the appropriate primary key columns of the correct data type.
- The syntax of all join conditions between the transaction and reference objects is correct.
- A reference object is not registered multiple times for an event class.
- A reference object is not assigned to another reference object.

See: Transaction Objects Introduction, page 3-1

Creation of Sources

To create sources, the Create and Assign Sources program verifies whether a source with the same source code for the application already exists.

Note: When creating sources, the Create and Assign Sources program does not perform the following tasks:

- Flagging sources as summed
- Providing user friendly names or descriptions for the sources
- Providing lookup types or value sets for the sources
- Marking the sources as Accounting Flexfield or Accounting Flexfield qualifiers

Update the source definitions appropriately.

Assign Sources

For each database object column, the Create and Assign Sources program verifies that the corresponding source is assigned to the transaction or reference object's event class.

- If no source assignment exists, the Create and Assign Sources program creates a new source assignment.

  The source assignment level (header, line, or reporting currency) depends on the transaction object type.

  The reference object inherits the extract object level of the source assignment in the joined transaction object.

  If the sources are generated from reference objects, the program can assign them to event classes even if the corresponding application is different from that of the sources.

Validate Only Mode

Select the Validate Only mode when launching the Create and Assign Sources from the Submit Request window or by selecting the **Validate Assignments** button in the Event Class Options page. When run in the Validate Only mode, the Create and Assign Sources program performs the following tasks:

- Validation of Transaction and Reference Objects Defined in the AMB, page 5-10
- Validation of Sources and Source Assignments, page 5-10
- Validation of Accounting Attribute Assignments, page 5-10
- Duplicate Column Definitions, page 5-11

Validation of Transaction and Reference Objects Defined in the AMB

The Create and Assign Sources program verifies the following for transactions and reference objects:

- All transaction and reference objects defined for the event class exist in the database data dictionary.
- All transaction objects, based on the transaction object type, contain the appropriate primary key columns of the correct data type.
- The syntax of all join conditions between the transaction and reference objects is correct.
- A reference object is not registered multiple times for an event class.
- A reference object is not assigned to another reference object.

See: Introduction to Transaction Objects, page 3-1

Validation of Sources and Source Assignments

The Create and Assign Sources program performs the following source and source assignment validations:

- A source assigned to an event class is defined in at least one transaction or reference object associated with the event class.
- The data type of the source matches the data type of a corresponding column from a transaction or a reference object assigned to an event class.
- The source is assigned to the event class at a level that is consistent with the transaction or reference object type.

Validation of Accounting Attribute Assignments

The Create and Assign Sources program validates all accounting attribute assignments for the event class with the following validations:

- Standard sources must be assigned to all accounting attributes that are required and do not belong to a group.
- If any of the accounting attributes belonging to a group are assigned to a standard source, then all the required accounting attributes for that group must also be assigned to the event class.

  For example, if one of the accounting attributes belonging to the group Third Party is assigned, then all other required accounting attributes for the same group must also be assigned.

- If the Encumbrance Accounting flag for the event class is selected, then the accounting attribute Encumbrance Type identifier must be assigned to a standard source for that event class.

- If the Budget Accounting flag for the event class is selected, then the accounting attribute Budget Version Identifier must be assigned to a standard source for that event class.

See: Revise Source Definitions and Assign Accounting Attributes, page 6-3

Duplicate Column Definitions

Since many transaction objects are assigned to an event class, the corresponding database objects may contain the same column definition many times. If a database object column definition is the same as that of another database object, the Create and Assign Sources program assumes that the columns always contain the same value. The program creates a single source that is assigned to an event class.

When the Create and Assign Sources program encounters duplicate database column definitions, it applies the following rules in the following order to determine from which database object the Accounting Program selects the source values:

1. Always populated transaction objects are used in preference to those that are not always populated.

   This rule ensures that the Accounting Program selects source values from database objects that are most likely to have not null values for it.

2. Header transaction objects are used in preference to line transaction objects. The source is assigned to the event class at the highest level.

   This rule ensures that sources are available to use in as many journal entry setups as possible.

6

Revise Source Definitions and Assign Accounting Attributes

Revise Source Definitions and Assign Accounting Attributes Overview

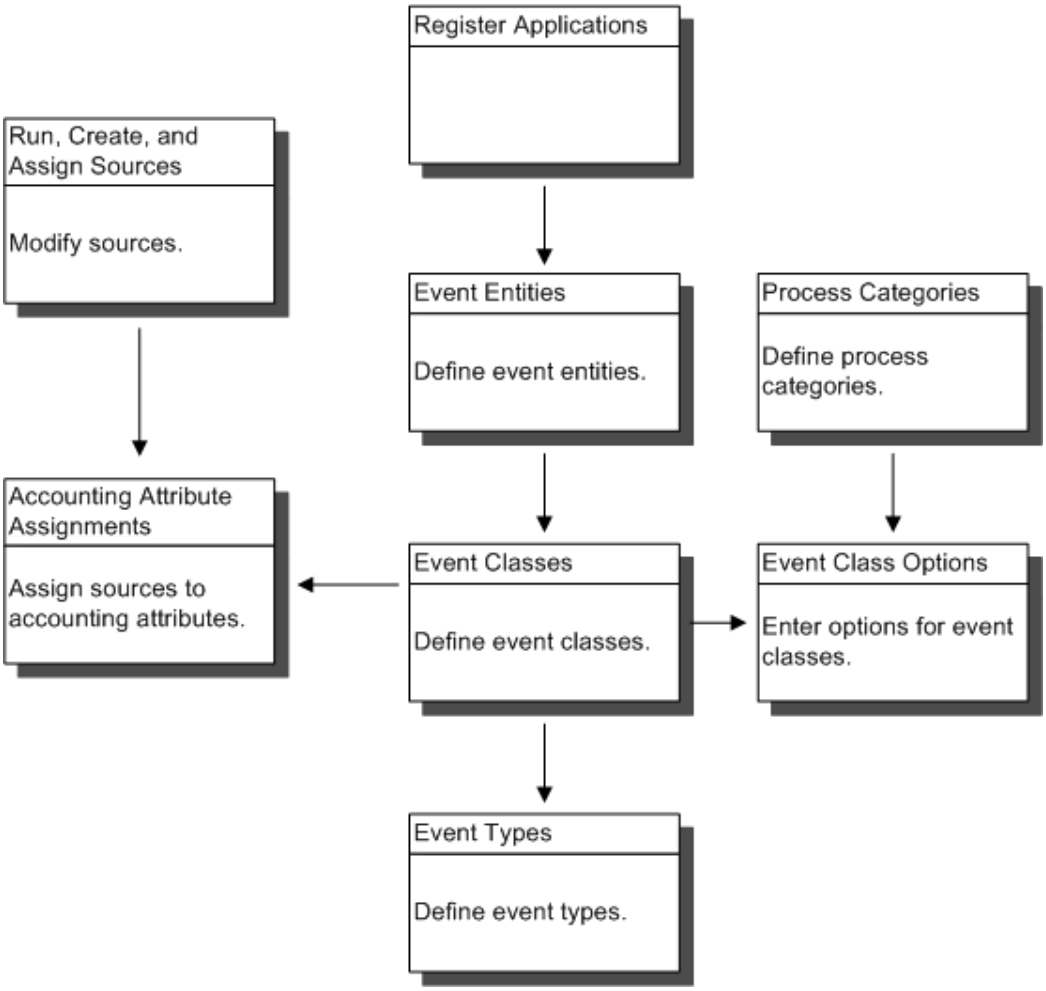
Once an application's sources are identified and set up, they must be seeded in the Accounting Methods Builder (AMB).

The Financial Services Accounting Hub also requires that sources be assigned to event classes. When standard sources are assigned to event classes, optionally assign them to accounting attributes. Certain accounting attributes always require an assignment.

This chapter provides guidelines on modifying sources and assigning sources to accounting attributes in the AMB. Information is also provided on naming standards for the sources stored in the transaction objects.

The diagram below provides an overview of the setup steps to seed sources and event information in the AMB and is described in Seeding Event Information Using Accounting Methods Builder Introduction, page 4-1.

Event Setup Overview



The following steps are described in Event Setup Steps, page 4-1:

- Register applications
- Event entities
- Event classes
- Event types
- Process categories
- Event class options

Revise Source Definitions and Assign Accounting Attributes

Once the Create and Assign Sources program is completed successfully, you can revise your source definitions.

Complete the following revisions before using the sources in journal setups or assigning them to accounting attributes:

- Sources that correspond to Accounting Flexfield identifiers must have a data type of Number, the Key Flexfield check box selected, the flexfield application set to General Ledger, and the flexfield title set to Accounting Flexfield.
- Sources that correspond to Accounting Flexfield qualifiers must have a data type of Alphanumeric and an Accounting Flexfield qualifier must be selected.
- Whenever appropriate, assign sources to lookup types or value sets.

See: Translated Sources, Lookup Types, and Value Sets Guideline, page 3-68

Complete the following revisions at any time:

- Source names should be consistent with the Financial Services Accounting Hub source naming standards.

  See: Standards for Source Names and Descriptions, page 6-3

- Sources that are rarely used in journal setups should not be displayed.

In addition, prior to defining journal line types, you should revise your accounting attribute assignments if needed.

See: To Assign Accounting Attributes in Accounting Methods Builder, page 6-15

Standards for Source Names and Descriptions

To set up appropriate application accounting definitions, users and implementers of the Financial Services Accounting Hub need to understand the origins, meaning, and context of sources. It is therefore important that naming standards be employed for source names and descriptions for the following reasons:

- Enables users to easily identify a source
- Ensures consistency in nomenclature between all subledger applications

At a minimum, follow the standards listed below for source names and their descriptions. These guidelines are subordinate to the consistency guideline that sources should be defined only once and assigned to all the classes that require them.

See: Consistency Guideline, page 3-70

Source Names

Define user-oriented source names composed of a user entity name plus a user name.

In Oracle applications, the user entity name is usually the name of a block on an Oracle applications window. For example, Invoice, Supplier, Supplier Site, and System Options are examples of user entity names for Payables. User entity names for Receivables include Transaction, Customer, Transaction Type, and Transaction Line.

The user name is generally the same as the field label displayed to users in a window or on a report.

> **Note:** Source names must not have underscores (_) or periods (.) between words. For example, do not use names like Invoice_Number or Invoice.Numbers.

Oracle Receivables Example: Assume that Receivables decides to include the GL date from the transactions window as a source. In this case, the user entity name is Transaction and the user name is GL Date. Therefore, the source name should be Transaction GL Date.

Oracle Inventory Example: If Inventory decides that they should include the shipment number from the material transactions window as a source, then the name for this source is Material Transactions Shipment Number.

> **Note:** Only approved abbreviations should be used in source names. If abbreviations are inconsistent across applications, it is difficult for users to understand the origin of a source. Also, source names must be unique by application.

Accounting Flexfields

Accounting Flexfield sources should be named as follows: <User Entity Name> + <Accounting Flexfield Name> + <Account>. If the Accounting Flexfield name already includes the word Account, then it should not be appended to the end of the source name.

Assets Example: The Accounting Flexfield in the Asset Category definitions window is called Asset Cost. This Accounting Flexfield has the source name Category Asset Cost Account.

Inventory Example: The Accounting Flexfield in the Item window in Oracle Inventory is called the Cost of Goods Sold Account. The source name is Item Cost of Goods Sold Account. Since the string Account is part of the flexfield name, it is not appended again.

Descriptive Flexfields

Descriptive flexfield sources are named as follows: <User Entity Name> + <Attribute #>,

where # is the number of the segment being named. This highlights the fact that each segment of a descriptive flexfield serves as a different source.

Oracle Payables Example: Payables has a descriptive flexfield entitled Invoice. In this case, the source name for the first attribute is Invoice Header Attribute 1. The name for the second attribute is Invoice Header Attribute 2.

Global Descriptive Flexfields

Global descriptive flexfields sources are named <User Entity Name> + <Global Attribute #>, where # is the number of the attribute being named.

Oracle Regional Localizations Example: The source name for the first attribute of the global descriptive flexfield JP_AP_AWT_TAX_RATES is Withholding Tax Rates Global Attribute 1.

Key Flexfields

Key flexfields sources corresponding to key flexfields that support multiple structures are named <User Entity Name> + <Segment #>, where # is the number of the segment being named. In this case, define a separate source for each key flexfield segment.

The word flexfield should be in the description, but it does not have to be in the name of the source. The exception is when users need the word flexfield to understand the source name.

Inventory Example: The source name for the first segment of the Stock Locator Key Flexfield is Stock Locator Key Segment 1.

For key flexfields that only support a structure, a single source to store the key flexfield combination identifier is required. The name of the source should match the key flexfield title.

Assets Example: The source name for the Asset Category Key Flexfield is Asset Category Flexfield.

Surrogate Keys (Technical Identifiers)

Sources that represent technical identifiers are named <User Entity Name> + <Identifier>.

Payables Example: The source name for the identifier INVOICE_DISTRIBUTION_ID is Invoice Distribution Identifier.

Assets Example: The source name for the identifier ASSET_ID is Asset Identifier.

> **Note:** This guideline does not apply to sources that have assigned lookups or value sets.

See: Translated Sources, Lookup Types, and Value Sets Guideline, page 3-68

Address Elements

Sources that represent elements of an address are named <User Entity Name> + <Address> + <Address Element>. However, if the element of the address is already called Address then do not repeat the <Address> part of the name.

Payables Example: The source name for the Country of a Supplier Site Address is Supplier Site Address Country.

Options and Indicators

Source names for options that are generally represented by check boxes in windows must include the word Option at the end of their name.

While options are typically regarded as being switches with either an on or off value, indicators are required fields or radio groups that the subledger application uses to obtain a value. The source name to represent this indicator must include the word Indicator at the end of the name.

Inventory Example: The source name to represent the Costing Enabled option for an inventory item is Item Costing Enabled Option.

Payables Example: In Payables, the amount of the distribution alone cannot be used to determine whether there is a gain or loss between the invoice and payment. Payables uses an indicator to flag the distribution as either a gain or a loss.

The source name to represent the gain or loss on invoice distributions is Gain or Loss Indicator.

System Sources

The names of system sources are reserved. Standard sources must not have the same name as system sources.

See: System Sources Guideline, page 3-74

Source Descriptions

The source description enables users to tie the source to its originating application field.

Flexfields

The description for key, global, or descriptive flexfields should include the flexfield title as it appears in the system administration application. This should be followed by the word Segment or Attribute plus the segment or attribute number. This standard is to ensure that users know easily which flexfield segment a source represents.

Assets Example: The description for the first segment of the Assets Category Flexfield is FA Category Flexfield Segment 1.

Internal (Unique) Identifiers

Sources that are unique identifiers must have user-oriented descriptions that include table and column names.

Unique identifiers are typically never displayed to users. However, because they can be useful in setting up certain application accounting definitions, some of these identifiers can be made available as sources. If made available, they should include descriptions allowing users to understand their nature as unique identifiers for a particular type of transaction or transaction line.

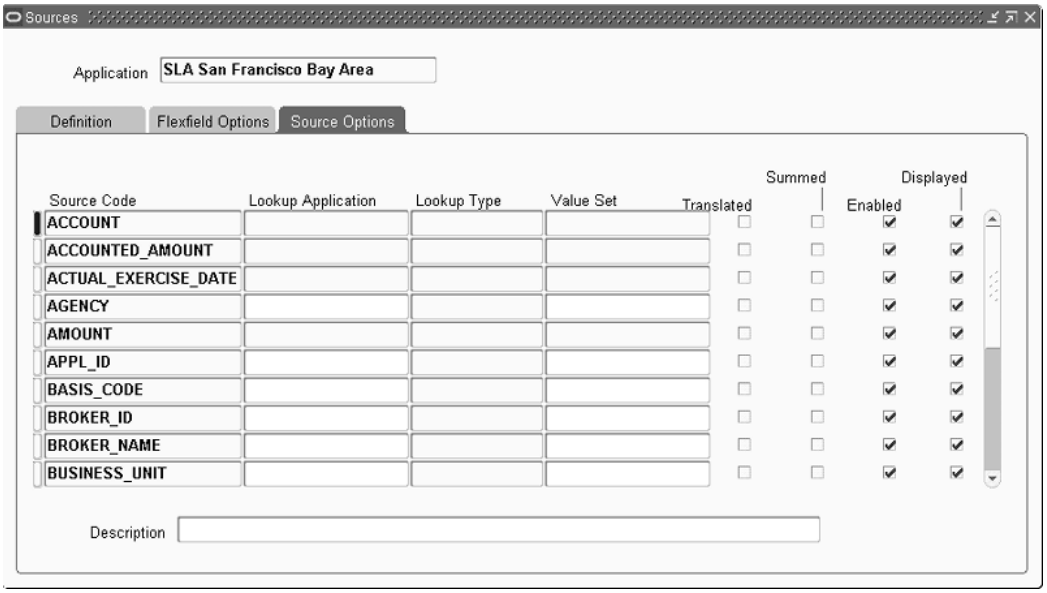
Payables Example: The Invoice Identifier for Payables could include a description such as the following:

Uniquely identifies Payables invoice transactions (AP.INVOICES.ALL INVOICE.ID)

Manually Defining/Revising Sources

Define sources for a particular application in the Sources window. These sources are part of the startup data for Financial Services Accounting Hub implementation.

To Manually Define/Revise Sources in Accounting Methods Builder



Definition Tab

The table below describes selected fields in the Definition tab of the Sources window.

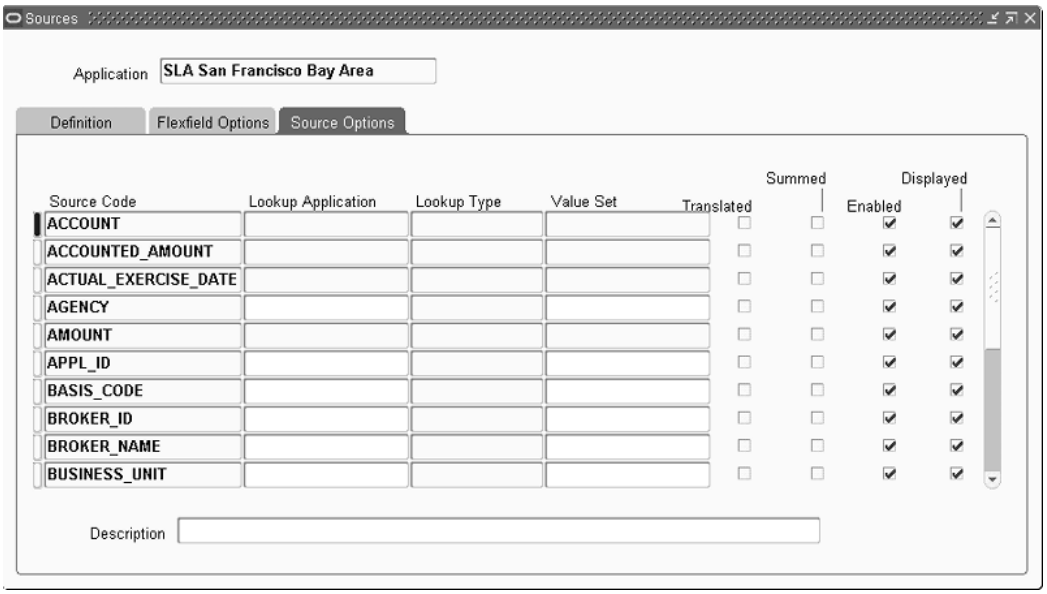
Selected Fields in the Sources Window, Definition Tab

Field	Description
Source Code	Identifies a source; must match the name of the corresponding transaction object column; used by programs and routines to refer to this particular source; used to track and troubleshoot problems associated with its definition. Sources defined using the Sources window are always standard sources. See: 6. Build Transaction Objects and Programs, page 1-12
Displayed Name	Source user name; appears in the list of values throughout the AMB
Data Type	Source data type. Valid values are Number, Alphanumeric, Accounting Flexfield, Integer, and Date. For sources created by the Create and Assign Sources program, the AMB automatically populates this field with the column data type; cannot be updated.

Source Options Tab

Use the Source Options tab to assign lookup types and value sets to sources and to define miscellaneous source properties.

See: Translated Sources, Lookup Types, and Value Sets Guideline, page 3-68



1. If the source does not have a value set associated with the lookup application and the Translated check box is not selected, in the Lookup Application field, select a lookup application from the list of values.

   The list of values includes all Oracle Applications.

2. If a value set is not selected and the Translated check box is not selected, in the Lookup Type field, select the lookup type to associate with the source from the list of values.

   The Accounting Program uses the lookup type to translate internal codes into user friendly names.

   Payables Example: An implementer creates a source with the display name Invoice Type and assigns a Lookup Type of INVOICE_TYPE to this source. The INVOICE_TYPE_LOOKUP_CODE is stored as a source value in the transaction object.

3. If a lookup type is not selected and the Translated check box is not selected, in the Value Set field, select the value set to associate with the source.

   The value set restricts the possible values for a source and translates internal codes into user friendly names.

   The list of values for this field contains value sets with any of the following validation types: Table, Independent, or Translatable Independent.

   See: Overview of Values and Value Sets, *Oracle Applications Flexfield Guide*

   Receivables Example: An implementer determines that the name of the payment terms associated with an invoice is made available as a standard source. When users define payment terms, they can use application object library MLS features to

record payment term names in many languages. For example, users can define a single payment term with the name IMMEDIATE in English and INMEDIATO in Spanish.

To avoid having to store the payment terms names in all installed languages in the transaction object, implementers can create a value set called Receivables Terms and assign it to the source. This value set is defined with RA_TERMS.NAME as the Value and RA_TERMS.TERM_ID as the identifier.

The TERM_ID is stored in the transaction object while the NAME is displayed to users. The table below lists a possible source definition.

Prompt	Value
Source Code	INVOICE_TERM_ID
Data Type	Number
Display Name	Invoice Terms Name
Description	Name of the Payments Terms assigned to the invoice
Value Set	Receivables Terms
Translated	No

Similar considerations apply to lookup codes; the source display name should refer to the meaning that is displayed to users and not the code.

4. If the source does not have a lookup type or value set associated with it, select the Translated check box to identify sources whose values are language dependent.

   If this option is selected, a value must be stored in the transaction object for each translated language. You cannot use translatable sources in conditions.

5. To identify numeric sources whose values can be treated like amounts, select the Summed check box.

   For example, quantities in Inventory and Purchasing may be eligible for summarization and can be marked as Summed. If this option is selected, multiple lines can be summarized for a source to produce a total, assuming that the source values is the same for the lines.

6. Select the Enabled check box to make this source available for use.

7. Select the Displayed check box to indicate that the source is available whenever a

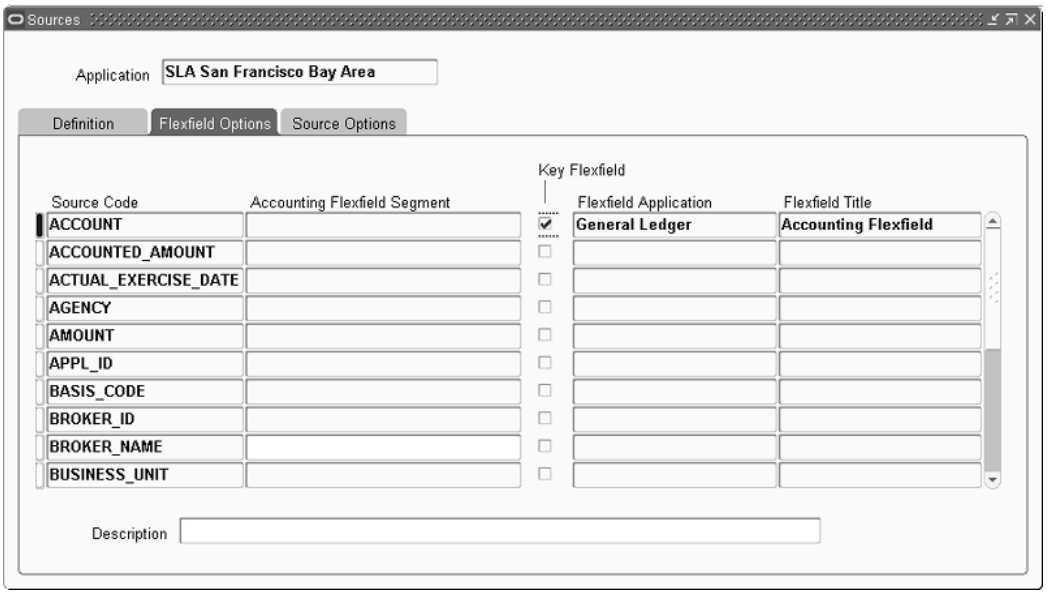
source list of values appears in the following windows in the AMB:

- Journal Line Types
- Account Derivation Rules
- Journal Entry Descriptions
- Supporting References

If the option is set to No, the list of values for the Source fields in all of the above windows excludes this source. Select this check box only for sources which you believe are most likely to be used in journal entry setups.

Flexfield Options Tab

Use the Flexfield Options tab to mark a standard source as a key flexfield and to provide other flexfield options.



Sources Window, Flexfield Options Tab

Field	Description
Accounting Flexfield Segment	If the application is an Oracle application, the list of values includes Accounting Flexfield qualifiers. If the application is a custom application, the list of values includes the Accounting Flexfield qualifiers and the value Other which enables users to mark a source as any segment of the Accounting Flexfield. This field cannot be updated once the source is created and is assigned to accounting attributes. If this field is populated, the following occurs: • The Key Flexfield check box is automatically selected. • The Flexfield Application field defaults to General Ledger. • The Flexfield Title field defaults to Accounting Flexfield. • The Accounting Flexfield Segment, Flexfield Application, and Flexfield Title fields are disabled.
Key Flexfield	Select this check box if the source data type is Number and the Accounting Flexfield Segment field is not populated to mark the source as a key flexfield. If this check box is selected, the Translated check box, the Summed check box, the Lookup Application field, the Lookup Types field, and the Value Set field are disabled; cannot be updated once the source is created and assigned to accounting attributes.

Field	Description
Flexfield Application	Select a flexfield application if the source is marked as a key flexfield and the Accounting Flexfield Segment field is not populated; cannot be updated once the source is created and assigned to accounting attributes. The list of values includes all applications registered in the Application Object Library that own key flexfields.
Flexfield Title	Select a flexfield title if the source is marked as a key flexfield and the Accounting Flexfield Segment field is not populated; cannot be updated once the source is created and assigned to accounting attributes. The list of values includes all the flexfield titles for the flexfield application selected and which supports one flexfield structure.

Assigning Sources

Sources are used in application accounting definitions to derive information used in creating subledger journal entries. To use sources in the AMB, sources must be assigned to event classes.

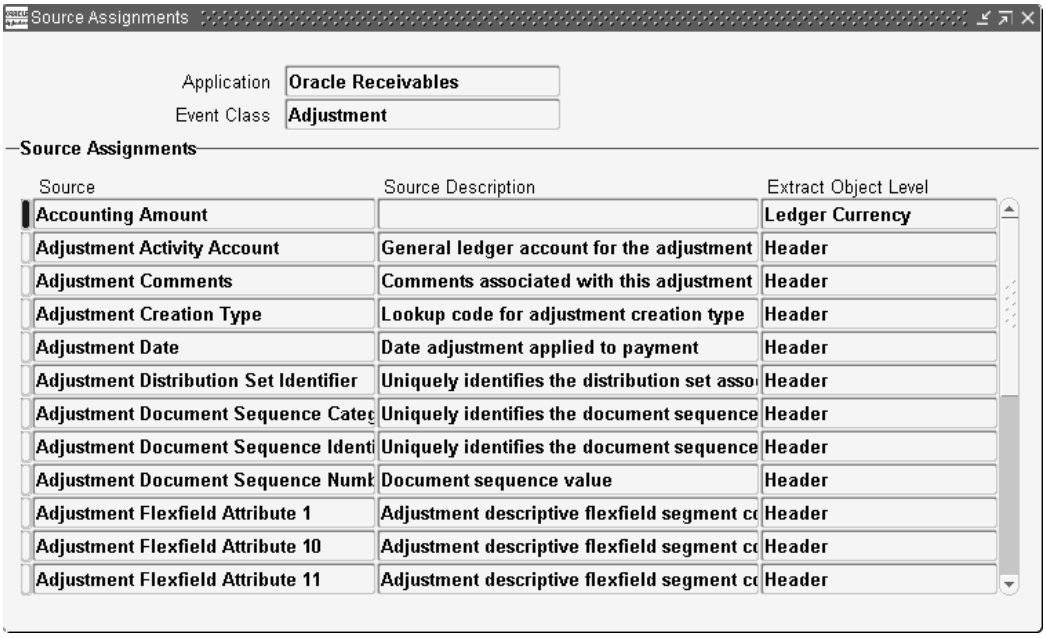
The purpose of a source assignment is to specify when a source is available for creating subledger journal entries.

Sources are only available for the event classes to which they are assigned. A source can be assigned to more than one event class. Source values are stored in the transaction or reference objects for each event belonging to the event class to which the source is assigned.

See:

- Sources and Source Data, page 1-3
- Introduction to Event Entities, Event Classes, and Event Types, page 4-3

To Assign Sources in the AMB

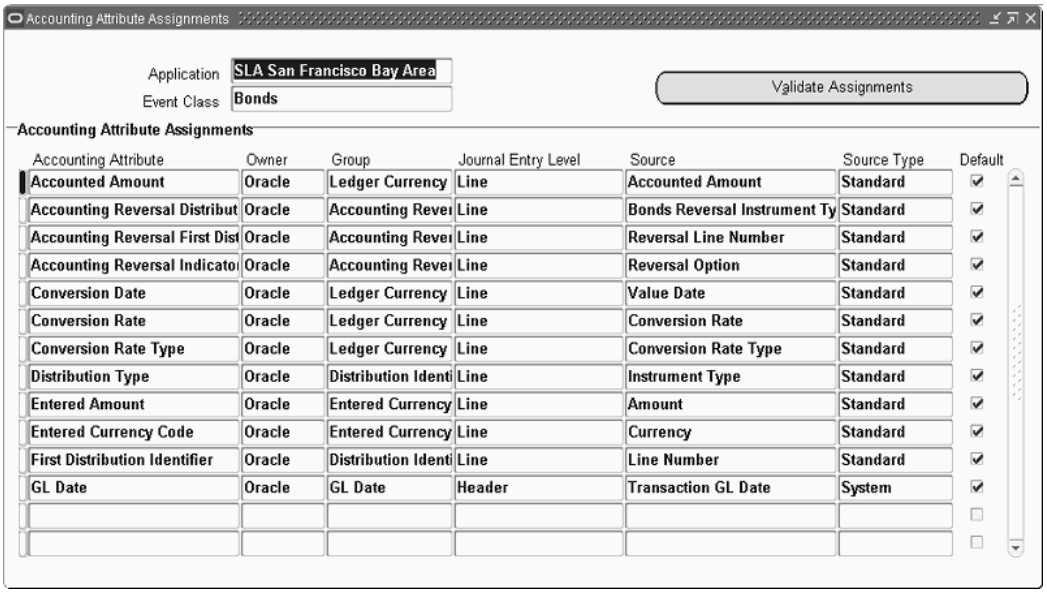


Selected Fields in the Source Assignments Window

Field	Description
Source	Source name that you are assigning to the event class The list of values includes all sources set up for this application with the following exception: • If the Displayed option in the Sources window, Source Options tab is set to No, then the list of values excludes that source.

Field	Description
Extract Object Level	Extract object level at which the source value is provided. The possible values are Header and Line. If the value of a source varies for each distribution line in the extract objects data for an event, the level for the source assignment is Line. If its value does not vary by distribution line, the level is Header.

To Assign Accounting Attributes in the AMB



Selected Fields in the Accounting Attribute Assignments Window

Field	Description
Accounting Attribute	Based on the accounting attribute selected, the Owner, Group, and Journal Entry Level fields are automatically populated. For components seeded by Oracle, the owner is Oracle. For components created on site by implementers, the owner is User.

Field	Description
Source	Select a source with a source type of standard, system, or custom sources for the assignment; can be assigned to one or more accounting attributes for an event class.
Source Type	Automatically populated
Default	Select to have the accounting attribute assignment default to the journal line types for line accounting attributes or to the application accounting definitions for header accounting attributes; always updateable. **Note:** Changing the default flag on an accounting attribute assignment will impact existing journal line types or application accounting headers that use the event class default. See: Accounting Attributes Guideline, page 3-17

7

Financial Services Accounting Hub Seed Data

Financial Services Accounting Hub Seed Data Overview

The Financial Services Accounting Hub provides the following types of seed data:

- Subledger application setups

  For these setups, FNDLOAD technology imports and exports the data between the database tables and the ldt data file directly.

  See: Financial Services Accounting Hub Setups, page 7-1

- Post-Accounting programs

  Subledger applications, such as Oracle Payables, support programs that transfer transaction data between subledgers based on the accounting generated by the transaction date. Users define Post-Accounting programs to distinguish journal lines for processing based on the accounting class associated with the journal entry line.

  See: Post-Accounting Programs, *Oracle Subledger Accounting Implementation Guide*

- Application accounting definitions and journal entry setups

  The Financial Services Accounting Hub provides the Application Accounting Definitions Loader (AAD Loader) to transfer setups between the database and the ldt file.

  See: Application Accounting Definitions (AAD) Loader Overview, page 17-1

Subledger Application Setups

The subledger application setups are the data created by the subledger applications to enable subledger accounting functionality. These setups establish the framework for

building application accounting definitions. The setups include the following objects:

- Subledger applications registered with the Financial Services Accounting Hub
- Event model
- Process categories
- Accounting event class options
- Sources
- Source assignments
- Accounting attribute assignments
- Custom sources

Specifications

FNDLOAD exports and imports the subledger accounting setups between the database and ldt file. Use the xlamseed.lct control file for this purpose.

Export

To export the subledger application setups from a database to an ldt data file, use the following command:

```
FNDLOAD <username>/<password>[@connect] 0 Y DOWNLOAD xlaemseed.lct
<datafile> XLA_SUBLEDGERS APPLICATION_ID=<application id>
```

Parameter	Description
Application ID	Application internal identifier

Import

To import the subledger application setups from an ldt data file to the database, use the following command:

```
FNDLOAD <username>/<password>[@connect] 0 Y UPLOAD xlaemseed.lct
<datafile>
```

FNDLOAD inserts and updates the subledger application setups in the database; however, no existing data is deleted from the database.

See: Subledger Application Clean-up Script, page 7-3

Subledger Application Setups Examples

Export Example

For Oracle Receivables to export the subledger application setups from the database to the ldt data file ARXLASD.LDT, run the following:

```
FNDLOAD scott/tiger 0 Y DOWNLOAD xlaemseed.lct ARXLASD.ldt
XLA_SUBLEDGERS APPLICATION_ID=222
```

Import Example

For Receivables to import the subledger application setups from the ldt data file ARXLASD.LDT to the database, run the following:

```
FNDLOAD scott/tiger 0 Y UPLOAD xlaemseed.lct ARXLASD.ldt
```

Subledger Application Setups Clean-up Script

The following script deletes the subledger application setups from the database. Once the data is deleted it is not recoverable unless the data was exported to an ldt data file.

> **Note:** It is not necessary to run this script before running the import process. Run this script only if data must be removed from the database.

XLA_AAD_LOADER_UTIL_PVT.purge_subledger_seed Parameters

Parameter	Description	Data Type	Type
P_API_VERSION	API version. The value should be 1.0	NUMBER	IN

Parameter	Description	Data Type	Type
X_RETURN_STATUS	API return status. Possible values are: • FND_API.G_RET_STATUS_SUCCESS - completed successfully • FND_API.G_RET_STS_ERROR - completed unsuccessfully • FND_API.G_RET_STS_UNEXP_ERROR - completed with unexpected error	VARCHAR2(1)	OUT
P_APPLICATIONS_ID	Internal identifier application	INTEGER	IN

Example

```
DECLARE

x_return_status VARCHAR2(1) ;

BEGIN

XLA_AAD_LOADER_UTIL_PVT.purge_subledger_seed
(p_api_version        => 1.0
,x_return_status      => x_return_status
,p_application_id     => <application id>) ;

END;
```

8

Seeding Calls to Subledger Event Capture Routines

Seeding Calls to Subledger Event Capture Routines Overview

The Financial Services Accounting Hub provides a set of common APIs to capture accounting events. All event operations must be performed through these APIs. By ensuring that event operations are executed through generic APIs, the architecture meets the following needs of implementers:

- Insulates implementers from changes in the implementation of an API

  The presence of the API reduces dependencies between the Financial Services Accounting Hub and the subledger applications.

  Implementers do not have to know the underlying Financial Services Accounting Hub data model to capture accounting events. In addition, any implementation changes made by the Financial Services Accounting Hub has minimum or no impact on implementers.

- Provides consistency in API calls across applications

  The common API performs all necessary validations and guarantees a consistent implementation across applications.

The capture and recording of accounting events is an essential task. Accounting events are usually captured and recorded when they occur. Additionally, application features allow transactions that have been recorded, but not yet accounted, to be modified, canceled, or deleted. The routines for capturing and recording accounting events must take into account all of these circumstances without reducing user functionality.

The Accounting Program selects accounting events based on criteria specified by users. The Accounting Program does not check for any functional dependencies between transactions or event types.

Note: For each eligible event, the Accounting Program retrieves source values from the transaction objects. Since these events are used to create subledger journal entries, a great deal of information must be reliably obtained from the transaction data model.

See: Transaction Objects Guidelines, page 3-14

Procedures to Seed Event Capture Routines

Implementers must undertake the following steps to seed calls to event capture routines:

- Perform Event Setups in Accounting Methods Builder, page 8-2
- Write Product Specific Cover Routines, page 8-2
- Integrate Event APIs with Subledger Applications, page 8-4

Perform Event Setups in Accounting Methods Builder (AMB)

This is a prerequisite step. Define and register event entities, event classes, and event types in the Accounting Methods Builder (AMB) before events can be captured. The event APIs use event information to perform the necessary validations when creating events.

See:

- Transaction Objects Analysis Introduction, page 3-1
- Event Setup Steps, page 4-1

Write Product Specific Cover Routines

Event APIs provided by the Financial Services Accounting Hub are generic public APIs available to all applications. These APIs have generic parameters to suit a common purpose. For example, each API has a security context parameter to capture the transaction security context for an application.

In order to reduce dependencies and facilitate maintenance, it is recommended that you write a wrapper routine on top of the Financial Services Accounting Hub APIs. Wrapper routines can encapsulate application-specific logic to perform necessary validations before creating new events. Instead of calling a generic API, call this wrapper from the subledger application to perform the necessary event operations. Map application-specific parameters to the API generic parameters.

For example, Receivables can have a package called AR_XLA_EVENTS_PKG, which contains all the APIs to implement accounting events in Receivables. The code for this

package is shown below.

Assume that Receivables has two event classes Receipts and Transactions. To handle accounting events for Receipts transactions, create a procedure **create_receipt_event()** with p_receipt_id as an input parameter, instead of using a generic parameters like source_id_int_1. The procedure **create_receipt_event()** calls the appropriate Financial Services Accounting Hub API to create an event.

Similarly, create **create_transaction_event()** to handle accounting events for the event class Transactions.

```
AR_XLA_EVENTS_PKG
-- Procedure to create events for Receivables receipts
PROCEDURE create_receipt_event
(p_receipt_id
,p_event_type
,p_event_date
,p_event_status) IS
l_receipt_source_info XLA_EVENTS_PUB_PKG.t_event_source_info;
l_security_context XLA_EVENTS_PUB_PKG.t_security;
BEGIN
-- Perform Product specific checks
...
-- Map generic event API parameter to the product specific columns
l_receipt_source_info.application_id = 202;
l_receipt_source_info.legal_entity_id = l_legal_entity_id;
l_receipt_source_info.source_id_int_1 = p_receipt_id;
-- Call XLA API
XLA_EVENTS_PUB_PKG.create_event
(p_event_source_info   =>  l_receipt_source_info
,p_event_type_code     =>  p_event_type
,p_event_date          =>  p_event_date
,p_event_status_code   =>  p_event_status
,p_event_number        =>  NULL
,p_referenece_info     =>  NULL
,p_security_context    =>  l_security_context);
...
...
EXCEPTIONS
....
END;
```

Integrate Event APIs with Subledger Applications

To capture accounting events, make the necessary changes to your module's base code to integrate with event APIs or subledger specific wrapper APIs. Wrapper APIs are written over the Financial Services Accounting Hub's event APIs.

This integration enables the application modules to create, update, and delete accounting events when there is a corresponding business event on the transaction.

Overview of Event APIs

The API's discussed in this chapter are listed below:

1. Get Event Information, page 8-4
   - Event Information
   - Event Status
   - Event Exists
2. Create Events, page 8-5
   - Single Event
   - Events in Bulk
3. Update Events, page 8-6
   - Event Status
   - Event Type, Event Date, Event Number, Event Reference Information
4. Delete Events, page 8-12
5. Update transaction Information, page 8-14
   - Transaction Number

The following sections describe these APIs and provide guidelines for their use in capturing accounting events. The Event API Details section, page 8-14 furnishes code details on the APIs.

Get Event Information

You may need to perform certain checks with respect to events. For example, before creating a new accounting event, it is necessary to check whether there is an existing unprocessed event for the same transaction with the same event type and event date.

You may also want to know the status of a particular event or query events already created for the transaction.

To perform these checks, obtain relevant event information.

To obtain all the events related to a transaction, do the following:

1. Determine the system transaction identifier and event class for the transaction.

   Note: System transaction identifiers identify the transaction on which transactions and events are based. The Financial Services Accounting Hub uses these identifiers to search the events table and identify all events related to this transaction.

   See: System Transaction Identifiers, page 2-8

2. Call the function Get_Array_Event_Info() with the appropriate transaction parameters. The function returns an array of all events for a transaction.

To obtain all the events created for a particular event type within a transaction, do the following:

1. Determine the system transaction identifiers, event class, and event type of the transaction.

2. Call the function Get_Array_Event_Info() with the appropriate transaction and event type input parameters. The function returns an array of all accounting events for that transaction and event type. Optionally pass the event class, event date, and event status to further restrict the rows returned.

To get information about a specific event, do the following:

1. Determine the event_id.

2. Call the function Get_Event_Info() with the event_id parameter. This function returns a PL/SQL record containing all information pertaining to that particular event.

The Event API Details section in this chapter provides information on the following APIs:

- The Get_Event_Status API returns the event status for a specified event.

- The Event_Exists API checks if an event exists for the specified criteria.

Create Events

This section describes the following guidelines on creating events using the create event APIs:

- Creating a Single Event, page 8-6

- Creating Events in Bulk, page 8-6

Creating a Single Event

To create a new event:

1. Determine the event type, event date, and event status for the new event.
2. Call the Financial Services Accounting Hub function Create_Event() with the appropriate input parameters.

The Create_Event() API creates a single event at a time. The function returns the event_id of the created event.

Creating Events in Bulk

Create events in bulk using the API Create_Bulk_Events().

> **Note:** Do not use this API for existing transactions that already have events associated with them. For performance reasons, bulk event APIs do not perform checks as to whether events for the transaction already exist. Therefore, use this API only to create events for new transactions that do not have any prior events created.

Update Events

Changes made to transactions can impact one or more events associated with the transaction. Update the event to keep the transaction data and related events synchronized.

Update an event as long as it is not processed. Once an event is processed, you cannot update the event or the data associated with it.

Use the following APIs to update your events:

- Update Event

  This is an overloaded API used to update the status, type, date, number, and reference information of a single event.

- Update Event Status

  This API updates multiple events to a given status.

Update Event Status

An event status reflects precisely where a transaction is in its accounting life cycle and what subsequent actions can be performed on the transaction data. Changes made to a transaction typically impacts the event status associated with the transaction. The new

state of the transaction is reflected in an updated event status.

For example, when validating an invoice in Payables, its invoice status becomes Validated. This change in status also has an impact on the event status associated with this transaction. In this case, if the event associated with the transaction has a status Incomplete, it needs to be updated to status Unprocessed to reflect the current state of the transaction.

See: Event Status, page 2-10

To change the status of a specific event:

1. Determine the event ID and event status to set.
2. Call the Financial Services Accounting Hub function Update_Event() with the appropriate input parameters.

   This API updates the status of a specific event.

To update the status of more than one event:

1. Determine the system transaction identifiers, event class, and event status to set.
2. Call the Financial Services Accounting Hub function Update_Event_Status() with the appropriate input parameters to update the status of all unprocessed events for the transaction.

(Update Event) Update Event Type

In some situations, it is necessary to change the event type of an event. For example, if changing an existing transaction results in a new event, either create the new event or change the event type of the existing event. Assume there are two event types defined in Payables: Invoice Validated and Invoice Adjusted. Note the following points:

- Any change made to the transaction after validation is considered to be an adjustment.
- There can be only one Invoice validated event per invoice.
- The event with the earliest GL date is the Invoice Validated event. The table below lists details along with corresponding event data of an invoice that is entered into the system and validated.

Line Num	Line Type	Amount	GL Date	Event #	Event Type	Event Status
1	Item	100	10-Jan-06	101	Invoice Validated	Unprocessed

After validating the invoice, the user adds a distribution dated 12-Jan-06. A new event is created Invoice Adjusted. The table below lists details of the corrected invoice along with corresponding event information.

Line Num	Line Type	Amount	GL Date	Event #	Event Type	Event Status
1	Item	100	10-Jan-06	101	Invoice Validated	Unprocessed
2	Item	300	12-Jan-06	102	**Invoice Adjusted**	Unprocessed

Now assume that the user changes the GL date on this new distribution from 12-Jan-06 to 08-Jan-06. The Invoice Adjusted event is changed to Invoice Validated since it now has a GL date earlier than the first line dated 10-Jan-06. Also, the original Invoice Validated event is updated to Invoice Adjusted as shown in the table below.

Line Num	Line Type	Amount	GL Date	Event #	Event Type	Event Status
1	Item	100	10-Jan-06	101	**Invoice Adjusted**	Unprocessed
2	Item	300	08-Jan-06	102	**Invoice Validated**	Unprocessed

(Update Event) Update Event Date

A transaction's GL date and the event date associated with it must always be the same. If the transaction date is changed, then you must update the event date to keep the event and transaction consistent. If the subledger application supports multiple GL dates, then depending on the transaction data, changing a transaction date can result in the creation, update, or deletion of an existing event.

Consider the following three examples to explain these scenarios. In all of these examples, there is a transaction with multiple distribution lines that is entered into the system and approved. The transaction results in events being created. The user then changes the date on one of the distribution lines.

Example 1

The table below lists details, along with corresponding event data, of an invoice that is entered into the system and validated.

Line Num	Event #	GL Date	Event Type	Event Status
1	101	10-Jan-06	Invoice Validated	Unprocessed
2	101	10-Jan-06	Invoice Validated	Unprocessed
3	101	10-Jan-06	Invoice Validated	Unprocessed

The user changes the GL date on line #3 from 10-Jan-06 to 11-Jan-06 and reapproves the invoice. In this example, note the following:

- There are other distribution lines associated with event #101 that are of the same event type.
- There are no events of the same type for the date 11-Jan-06. (Processed events are not considered.)

Since there is no event of the same event type for the new GL date with an event status of Unprocessed, create a new event for this transaction line. The table below lists details of the invoice with the new event.

Line Num	Event #	GL Date	Event Type	Event Status
1	101	10-Jan-06	Invoice Validated	Unprocessed
2	101	10-Jan-06	Invoice Validated	Unprocessed
3	**102**	**11-Jan-06**	**Invoice Adjusted**	Unprocessed

Example 2

The table below lists details along with corresponding event data of two invoices that are entered into the system and validated.

Line Num	Event #	GL Date	Event Type	Event Status
1	102	11-Jan-06	Invoice Validated	Unprocessed

The user changes the GL date on line #1 from 10-Jan-06 to 11-Jan-06 and revalidates the invoice. In this example, note the following.

- There are no other distribution lines associated with event #101 that are of the same event type for 10-Jan-06.
- There is another event 102 of the same type for the date 11-Jan-06 with status Unprocessed.

Since there is an event of the same event type for the new GL date and with event status Unprocessed, delete event #101. Event #102 now represents transaction data for both distribution lines.

The table below lists details on Event #102.

Line Num	Event #	GL Date	Event Type	Event Status
1	**102**	**11-Jan-06**	**Invoice Validated**	Unprocessed

Example 3

The table below lists details, along with corresponding event data, of an invoice that is entered into the system and validated.

Line Num	Event #	GL Date	Event Type	Event Status
1	101	10-Jan-06	Invoice Validated	Unprocessed

The user changes the GL date on line #1 from 10-Jan-06 to 11-Jan-06 and revalidates the invoice. In this example, note the following:

- There are no other distribution lines associated with event #101 that are of the same event type.
- There are no events of the same type for the date 11-Jan-06.

Since there is no event of the same event type for the new GL date having an event

status Unprocessed, do one of the following.

1. Update Event #101 to the new date. The table below lists details on the updated event #101.

Line Num	Event #	GL Date	Event Type	Event Status
1	**101**	**11-Jan-06**	**Invoice Validated**	Unprocessed

2. Delete event #101 and create a new event #102 with date 11-Jan-06. The table below lists details on this new event #102.

Line Num	Event #	GL Date	Event Type	Event Status
1	**102**	**11-Jan-06**	**Invoice Validated**	Unprocessed

(Update Event) Update Event Number

Within a transaction, the Accounting Program processes events by event number. Along with other event information, the event number must be passed to the API at the time the event is created. If the API does not receive the event number, then the Create_Event() API inputs the next maximum event number for the transaction.

Once an event is created, a need may arise to change the event number because of changes taking place in the transaction. You can use the API Update_Event() to change an event's number. The ability to update event number gives you more control of the order in which events for a transaction are accounted.

Consider the following example to explain the behavior in which events are processed depending on event number. A transaction has events E1, E2 and E3 with event numbers 1, 3 and 5 respectively. Events E1 and E2 are in status Processed while E3 is Unprocessed.

The table below lists the name, number, and status of the transaction's events.

Event Name	Event Number	Event Status
E1	1	Processed
E2	3	Processed

Event Name	Event Number	Event Status
E3	5	Unprocessed

Now, assume that two more events E4 and E5 are created with event numbers 2 and 6. The table below lists the name, number, and status of these new events.

Event Name	Event Number	Event Status
E4	2	Unprocessed
E5	6	Unprocessed

The Accounting Program does not examine any events that have already been processed. However, the next accounting cycle processes the events in the order E4, E3, and E5.

To avoid the confusion involved in cases like this, it is strongly recommended that applications do not create any events with an event number less than the maximum event number of processed events. In the above example, the maximum event number of the processed events was 3, so event E4 should not have received an event number of 2.

> **Note:** Do not attempt to create duplicate event numbers within a transaction as the API raises an exception during processing.

Delete Events

An event must be deleted if there is no transaction associated with it as when a transaction is deleted. In some cases, changing a GL date on a transaction might require deleting an event. Consider the following example.

The table below lists details, along with corresponding event data, of an invoice that is entered into the system and validated.

Line Num	Event #	GL Date	Event Type	Event Status
1	101	10-Jan-06	Invoice Validated	Unprocessed

Line Num	Event #	GL Date	Event Type	Event Status
2	101	10-Jan-06	Invoice Validated	Unprocessed
3	102	11-Jan-06	Invoice Validated	Unprocessed

The user then changes the GL date on line #3 from 11-Jan-06 to 10-Jan-06 and revalidates the invoice. In this example, note the following:

- There are other distribution lines associated with 10-Jan-06 that are of the same event type as event #102.
- There are no other distribution lines for event #102 of the same event type for the date 11-Jan-06, excluding Processed events.

Event #102 should be deleted, since there are no remaining lines associated with it. The table below lists details on the updated invoice along with corresponding event information.

Line Num	Event #	GL Date	Event Type	Event Status
1	101	10-Jan-06	Invoice Validated	Unprocessed
2	101	10-Jan-06	Invoice Validated	Unprocessed
3	**101**	**10-Jan-06**	**Invoice Validated**	Unprocessed

Call the Delete_Event() API to delete this event.

To delete all Unprocessed events associated with a transaction:

1. Determine the transaction source information.
2. Call the Financial Services Accounting Hub function Delete_Events() with the transaction source information.
3. Optionally, specify the event type, event status, or event date, to restrict the events deleted.

If a transaction is deleted, the Delete_Entity() API must be called to delete the row in the

XLA_TRANSACTIONS_ENTITY table.

Update Transaction Number

In the case where the transaction number of the transaction has been changed, this API updates the transaction number on the events in the XLA_TRANSACTION_ENTITIES table so that they are consistent with the transaction number on the transaction.

The API checks the source information for a valid application, legal entity, event entity, and source identifiers. However, no validation is performed on the transaction number.

Event API Details

This section describes the event APIs accessible by implementers to perform event operations. The APIs described are generic and available to all applications.

Event APIs have the following characteristics:

- Event APIs do not issue any commits.
- If an API fails, any changes made are rolled back and a standard exception is raised.
- All parameters are read only (IN parameters); there are not OUT parameters.
- An event date is always truncated.

  The Financial Services Accounting Hub does not store the timestamp for an event date.
- All functions return a single value or record.

  The exception to this rule are the functions prefixed with a Get_Array string. For example, the function Get_Array_Event_Info() returns an array.
- Input parameters must be passed in a nonpositional notation, named notation.
- All the APIs called in query mode locks the event record in NOWAIT mode.

Event APIs have the following types of input parameters:

- System transaction identifiers

  These parameters capture information such as invoice_id or receipt_id. This information is stored with each event to later identify the source transaction for the event. You need to pass the source identifiers when creating an event.
- Transaction security identifiers

  Accounting events are subject to the security of the corresponding transaction. Every accounting event is stamped with its related transaction's security context.

Transaction security parameters capture application-specific transaction security information, such as operating unit or asset book.

Security information is required to perform any event operation. Event APIs ensure that security context information is passed for each event. It is your responsibility to pass the correct security context.

- Transaction reference information

  The reference parameters enable you to capture any miscellaneous reference or contextual information for an event. This information is optional and no validations are performed against any reference parameters.

The XLA_EVENT_PUB_PKG package is defined by the Financial Services Accounting Hub. This package contains the following items:

- PL/SQL record and table structures for common parameters
- CONSTANTS for event statuses

Use these constants and structures when passing and reading values to and from the APIs. This is done to keep implementers shielded from the implementation of events.

See:

- PL/SQL Data Types, page 8-33
- Constants, page 8-37

Get Event Information APIs

This section provides details on the APIs that obtain event information.

1. XLA_EVENTS_PUB_PKG.GET_EVENT_INFO()

Summary

```
FUNCTION XLA_EVENTS_PUB_PKG.get_event_info
(p_event_source_info  IN  xla_events_pub_pkg.t_event_source_info
,p_event_id           IN  INTEGER
,p_valuation_method   IN  VARCHAR2
,p_security_context   IN  xla_events_pub_pkg.t_security)
RETURN xla_events_pub_pkg.t_event_info;
```

Description

This API returns information for a particular event. The event is identified by specifying the transaction and event identifier. The API locks the specified event before returning any event information.

Validations

The API checks all source information for valid application, legal entity, event entity, and source identifiers (IDs). It ensures that the required parameters are not passed as null and that the event ID belongs to the same transaction, as the other transaction information being passed.

Parameters

See: Common Parameters, page 8-28

Return Value

The API returns a PL/SQL record containing event information.

See: PL/SQL Data Types, page 8-33

2. XLA_EVENTS_PUB_PKG.GET_ARRAY_EVENT_INFO()

Summary

```
FUNCTION XLA_EVENTS_PUB_PKG.get_array_event_info
(p_event_source_info   IN    xla_events_pub_pkg.t_event_source_info
,p_event_class_code    IN    VARCHAR2 DEFAULT NULL
,p_event_type_code     IN    VARCHAR2 DEFAULT NULL
,p_event_date          IN    DATE DEFAULT NULL
,p_event_status_code   IN    VARCHAR2 DEFAULT NULL
,p_valuation_method    IN    VARCHAR2
,p_security_context    IN    xla_events_pub_pkg.t_security)
RETURN xla_events_pub_pkg.t_array_event_info;
```

Description

This routine returns information for one or more events within a transaction. The calling program specifies the transaction and event selection criteria. Return information contains data on all events that belong to the specified transaction and fall under the given criteria. The API locks the specified events before returning the event information.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. It ensures that the required parameters are not passed as null and also validates the event class, event type, and event status. Note that the API truncates the event date.

Parameters

See: Common Parameters, page 8-28

Return Value

The API returns an array of event information.

See: PL/SQL Data Types, page 8-33

3. XLA_EVENTS_PUB_PKG.GET_EVENT_STATUS()

Summary

```
FUNCTION XLA_EVENTS_PUB_PKG.get_event_status
(p_event_source_info  IN   xla_events_pub_pkg.t_event_source_info
,p_event_id           IN   INTEGER
,p_valuation_method   IN   VARCHAR2
,p_security_context   IN   xla_events_pub_pkg.t_security)
RETURN VARCHAR2;
```

Description

This API returns the event status for a specified event. The calling program needs to specify the transaction and event identifier. The API locks the specified event record before returning the status.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. It ensures that the required parameters are not null and the event belongs to the same transaction as the other transaction information being passed.

Parameters

See: Common Parameters, page 8-28

Return Value

This API returns an event status. The Financial Services Accounting Hub has defined all event statuses as Constants.

See: Constants, page 8-37

4. XLA_EVENTS_PUB_PKG.EVENT_EXISTS()

Summary

```
FUNCTION XLA_EVENTS_PUB_PKG.event_exists
(p_event_source_info  IN   xla_events_pub_pkg.t_event_source_info
,p_event_class_code   IN   VARCHAR2 DEFAULT NULL
,p_event_type_code    IN   VARCHAR2 DEFAULT NULL
,p_event_date         IN   DATE DEFAULT NULL
,p_event_status_code  IN   VARCHAR2 DEFAULT NULL
,p_event_number       IN   INTEGER DEFAULT NULL
,p_valuation_method   IN   VARCHAR2
,p_security_context   IN   xla_events_pub_pkg.t_security)
RETURN BOOLEAN;
```

Description

This API checks whether an event exists for the specified criteria. It returns True if it finds at least one event matching the criteria; otherwise, it returns False. The API locks the event rows before returning a value.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. It ensures that the required parameters are not null and also validates the event class, event type, and event status. The API truncates the event date.

Parameters

See: Common Parameters, page 8-28

Return Value

The API returns True if an event is found for the specified criteria; otherwise, it returns False.

Create Event APIs

This section provides details on the APIs that create events.

1. XLA_EVENTS_PUB_PKG.CREATE_EVENT()

Summary

```
FUNCTION XLA_EVENTS_PUB_PKG.create_event
(p_source_event_info  IN  xla_events_pub_pkg.t_event_source_info
,p_event_type_code    IN  VARCHAR2
,p_event_date         IN  DATE
,p_event_status_code  IN  VARCHAR2
,p_event_number       IN  INTEGER DEFAULT NULL
,p_reference_info     IN  xla_events_pub_pkg.t_event_reference_info
DEFAULT NULL
,p_valuation_method   IN  VARCHAR2
,p_transaction_date   IN  DATE  DEFAULT NULL
,p_security_context   IN  xla_events_pub_pkg.t_security
,p_budgetary_control_flag  IN VARCHAR2)
RETURN INTEGER;
FUNCTION XLA_EVENTS_PUB_PKG.create_event
(p_event_source_info  IN  xla_events_pub_pkg.t_event_source_info
,p_event_type_code    IN  VARCHAR2
,p_event_date         IN  DATE
,p_event_status_code  IN  VARCHAR2
,p_event_number       IN  INTEGER DEFAULT NULL
,p_transaction_date   IN DATE DEFAULT NULL
```

```
,p_reference_info     IN  xla_events_pub_pkg.t_event_reference_info
DEFAULT NULL
,p_valuation_method   IN  VARCHAR2
,p_security_context   IN  xla_events_pub_pkg.t_security)
RETURN INTEGER;
```

Description

This API creates a new event. This function has two specifications, one with the p_budgetary_control flag and another without it.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. It ensures that the required parameters are not null and also validates the event type and event status.

No validations are performed against the reference columns and event number. However, if no event number is passed, the routine populates the next highest event number for that transaction. The event date is truncated.

Parameters

See: Common Parameters, page 8-28

Return Value

If an event is created successfully, then the function returns its event ID.

2. XLA_EVENTS_PUB_PKG.CREATE_BULK_EVENTS()

Summary

```
Procedure XLA_EVENTS_PUB_PKG.create_bulk_events
(p_source_application_id       IN INTEGER DEFAULT NULL
,p_application_id           IN  INTEGER
,p_legal_entity_id          IN  INTEGER DEFAULT NULL
,p_ledger_id                IN  INTEGER
,p_entity_type_code         IN  VARCHAR2);
```

Description

This API creates multiple events for multiple transactions. Information required for each event is inserted into the XLA_EVENTS_INT_GT table as described below, before the API is called.

XLA_EVENTS_INT_GT

Column Name	Data Type	Size	Required
ENTITY_CODE	VARCHAR2	30	Yes
APPLICATION_ID	NUMBER	15	Yes
LEDGER_ID	NUMBER	15	Yes
EVENT_STATUS_CODE	VARCHAR2	30	Yes
EVENT_TYPE_CODE	VARCHAR2	30	Yes
EVENT_DATE	DATE	-	Yes
TRANSACTION_DATE	DATE	-	No
VALUATION_METHOD	VARCHAR2	30	No
TRANSACTION_NUMBER	VARCHAR2	240	No
BUDGETARY_CONTROL_FLAG	VARCHAR2	1	No
SOURCE_ID_INT_1	NUMBER	15	No
SOURCE_ID_INT_2	NUMBER	15	No
SOURCE_ID_INT_3	NUMBER	15	No
SOURCE_ID_INT_4	NUMBER	15	No
SOURCE_ID_CHAR_1	VARCHAR2	30	No
SOURCE_ID_CHAR_2	VARCHAR2	30	No

Column Name	Data Type	Size	Required
SOURCE_ID_CHAR_3	VARCHAR2	30	No
SOURCE_ID_CHAR_4	VARCHAR2	30	No
SECURITY_ID_INT_1	NUMBER	15	No
SECURITY_ID_INT_2	NUMBER	15	No
SECURITY_ID_INT_3	NUMBER	15	No
SECURITY_ID_CHAR_1	VARCHAR2	30	No
SECURITY_ID_CHAR_2	VARCHAR2	30	No
SECURITY_ID_CHAR_3	VARCHAR2	30	No
REFERENCE_NUM_1/REFERENCE_NUM_4	NUMBER	15	No
REFERENCE_CHAR_1/REFERENCE_CHAR_4	VARCHAR2	30	No
REFERENCE_DATE_1/REFERENCE_DATE_4	DATE	-	No

Validations

The API checks all source information for valid application, legal entity, event entity, event number, and source IDs. It ensures that the required parameters are not null and also validates the event type and event status.

No validations are performed against the reference columns and event number.

Parameters

See: Common Parameters, page 8-28

Update Event APIs

- The Update_Event_Status() API updates the event statuses of more than one event of a transaction.

There is a set of overloaded APIs that can be used to update more than one attribute for an event. These APIs update event type, event date, event status, event number, and reference information for an event. All of these API's use the Update_Event() API.

> **Note:** Though these update event APIs retain the same name Update_Event(), they use the PL/SQL feature of overloading to create unique procedures. In overloading, the input parameter names and types are distinct, resulting in unique procedures. Different columns are updated in the tables, depending on which procedure is called.

For information on using overloading to create different procedure signatures, refer to the *PL/SQL User Guide*.

As discussed earlier, an event can be updated as long as it is not processed. Once an event is processed, you cannot update the event or the data associated with it.

This section provides details on the APIs that update events.

1. XLA_EVENTS_PUB_PKG.UPDATE_EVENT_STATUS()

Summary

```
PROCEDURE XLA_EVENTS_PUB_PKG.update_event_status
(p_event_source_info   IN  xla_events_pub_pkg.t_event_source_info
,p_event_class_code    IN  VARCHAR2 DEFAULT NULL
,p_event_type_code     IN  VARCHAR2 DEFAULT NULL
,p_event_date          IN  DATE DEFAULT NULL
,p_event_status_code   IN  VARCHAR2
,p_valuation_method    IN  VARCHAR2
,p_security_context    IN  xla_events_pub_pkg.t_security);
```

Description

This API updates the event status of one or more events within a transaction matching the specified criteria.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. It ensures that the required parameters are not null. The API also validates event status and if passed, event class and event type. The event date is truncated.

Parameters

See: Common Parameters, page 8-28

2. XLA_EVENTS_PUB_PKG.UPDATE_EVENT()

Summary

```
PROCEDURE XLA_EVENTS_PUB_PKG.update_event
(p_event_source_info   IN  xla_events_pub_pkg.t_event_source_info
,p_event_id            IN  INTEGER
,p_event_type_code     IN  VARCHAR2 DEFAULT NULL
,p_event_date          IN  DATE DEFAULT NULL
,p_event_status_code   IN  VARCHAR2 DEFAULT NULL
,p_event_number        IN INTEGER
,p_reference_info      IN
xla_events_pub_pkg.t_event_reference_info
,p_valuation_method    IN  VARCHAR2
,p_security_context    IN  xla_events_pub_pkg.t_security);
,p_transaction_date    IN  DATE DEFAULT NULL)
```

Description

This API updates multiple attributes of a single event. Using this API, the calling program can update event type, event date, and event status. An error code is returned if the update fails.

Validations

This API checks all source information for valid application, legal entity, event entity, and source IDs. The API ensures that the event ID is not null, that it belongs to the same transaction as the other transaction information being passed, and that the event has not already been accounted.

The parameters event type, event date, and event status are also validated if passed Not Null.

See: Common Parameters, page 8-28

3. XLA_EVENTS_PUB_PKG.UPDATE_EVENT()

Summary

```
PROCEDURE XLA_EVENTS_PUB_PKG.update_event
(p_event_source_info   IN  xla_events_pub_pkg.t_event_source_info
,p_event_id            IN  INTEGER
,p_event_type_code     IN  VARCHAR2 DEFAULT NULL
,p_event_date          IN  DATE DEFAULT NULL
,p_event_status_code   IN  VARCHAR2 DEFAULT NULL
,p_event_number        IN  INTEGER
```

```
,p_valuation_method     IN  VARCHAR2
,p_transaction_date     IN  DATE DEFAULT NULL
,p_security_context     IN  xla_events_pub_pkg.t_security);
```

Description

This API updates multiple attributes of a single event. Using this API, the calling program can update event type, event date, event status, and event number. An error code is returned if the update fails.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. No validations are performed against the event number but if no event number is passed, the routine populates the next highest event number for that transaction. The event date is truncated.

The API ensures that the event ID is not null, that it belongs to the same transaction as the other transaction information being passed, and that the event has not already been accounted. The parameters event type, event date, and event status are also validated if passed Not Null.

Parameters

See: Common Parameters, page 8-28

4. XLA_EVENTS_PUB_PKG.UPDATE_EVENT()

Summary

```
PROCEDURE XLA_EVENTS_PUB_PKG.update_event
(p_event_source_info    IN  xla_events_pub_pkg.t_event_source_info
,p_event_id             IN  INTEGER
,p_event_type_code      IN  VARCHAR2 DEFAULT NULL
,p_event_date           IN  DATE DEFAULT NULL
,p_event_status_code    IN  VARCHAR2 DEFAULT NULL
,p_reference_info       IN  xla_events_pub_pkg.t_event_reference_info
,p_valuation_method     IN  VARCHAR2
,p_transaction_date     IN  DATE DEFAULT NULL
,p_security_context     IN  xla_events_pub_pkg.t_security);
```

Description

This API updates multiple attributes of a single event. Using this API, the calling program can update event type, event date, event status, and the event's reference information. An error code is returned if the update fails.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. No validations are performed on the reference information.

The API ensures that the event ID is not null, that it belongs to the same transaction as the other transaction information being passed, and that the event has not already been accounted. The parameters event type, event date, and event status are also validated if passed Not Null.

Parameters

See: Common Parameters, page 8-28

5. XLA_EVENTS_PUB_PKG.UPDATE_EVENT()

Summary

```
PROCEDURE XLA_EVENTS_PUB_PKG.update_event
(p_event_source_info   IN  xla_events_pub_pkg.t_event_source_info
,p_event_id            IN  INTEGER
,p_event_type_code     IN  VARCHAR2 DEFAULT NULL
,p_event_date          IN  DATE DEFAULT NULL
,p_event_status_code   IN  VARCHAR2 DEFAULT NULL
,p_event_number        IN  INTEGER
,p_reference_info      IN  xla_events_pub_pkg.t_event_reference_info
,p_valuation_method    IN  VARCHAR2
,p_security_context    IN  xla_events_pub_pkg.t_security);
```

Description

This API updates multiple attributes of a single event. Using this API, the calling program can update event type, event date, event status, event number, and the event's reference information. An error code is returned if the update fails.

Note: This API updates both the event's event number and reference information.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. The API ensures that the event ID is not null, that it belongs to the same transaction as the other transaction information being passed, and that the event has not already been accounted.

The parameters event type, event date, and event status are also validated if passed Not Null. No validations are performed against the event number and reference information, but if no event number is passed, the routine populates the next highest event number for that transaction.

Parameters

See: Common Parameters, page 8-28

6. XLA_EVENTS_PUB_PKG.UPDATE_BULK_EVENT_STATUSES

Summary

```
Procedure XLA_EVENTS_PUB_PKG.update_bulk_event_statuses
(p_application_id           IN  INTEGER);
```

Description

This API updates the event status of multiple events. Before calling this API, users must populate the XLA_EVENTS_INT_GT table with the following:

- application_id
- entity_code
- ledger_id
- event_id
- event_status_code

The API updates the events in the XLA_EVENTS table to the new status.

Validations

This API validates the application ID, event entity, event ID, and event status. The status of both the new and old status cannot be Processed. The new status must be a valid event status.

Parameters

See: Common Parameters, page 8-28

Delete Event APIs

This section provides details on the APIs that delete events.

1. XLA_EVENTS_PUB_PKG.DELETE_EVENT()

Summary

```
PROCEDURE XLA_EVENTS_PUB_PKG.delete_event
(p_event_source_info   IN  xla_events_pub_pkg.t_event_source_info
,p_event_id            IN  INTEGER
,p_valuation_method    IN  VARCHAR2
,p_security_context    IN  xla_events_pub_pkg.t_security);
```

Description

This API deletes an unaccounted event based on its event identifier. The API returns an error code if the delete fails.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. The API also ensures that the mandatory parameters are not null, that

the event ID belongs to the same transaction as the other transaction information being passed, and that the event has not been accounted.

Parameters

See: Common Parameters, page 8-28

2. XLA_EVENTS_PUB_PKG.DELETE_EVENTS()

Summary

```
FUNCTION XLA_EVENTS_PUB_PKG.delete_events
(p_event_source_info   IN  xla_events_pub_pkg.t_event_source_info
,p_event_class_code    IN  VARCHAR2 DEFAULT NULL
,p_event_type_code     IN  VARCHAR2 DEFAULT NULL
,p_event_date          IN  DATE DEFAULT NULL
,p_valuation_method    IN  VARCHAR2
,p_security_context    IN  xla_events_pub_pkg.t_security)
RETURN INTEGER;
```

Description

This API deletes all events for a transaction that meet the specified criteria. When called, events that belong to the given event class, event type, and event date are deleted.

Validations

The API checks all source information for valid application, legal entity, event entity, and source IDs. It ensures that the required parameters are not null and if passed, validates the event type and event status.

Parameters

See: Common Parameters, page 8-28

Return Value

The function returns the number of events deleted.

3. XLA_EVENTS_PUB_PKG.DELETE_ENTITY()

If a transaction is deleted, users must call the DELETE_ENTITY API to delete the row in the XLA_TRANSACTION_ENTITIES table.

Summary

```
Function XLA_EVENTS_PUB_PKG.delete_entity
(p_source_info        IN  xla_events_pub_pkg.t_event_source_info
,p_valuation_method   IN VARCHAR2
,p_security_context   IN xla_events_pub_pkg.t_security)
RETURN INTEGER;
```

Description

This API deletes a row from the XLA_TRANSACTION_ENTITIES table. The routine checks if there are still events associated with the transaction. If yes, the routine does nothing and returns 1; otherwise, it deletes the transaction in the XLA_TRANSACTION_ENTITIES table and returns 0.

Validations

There are no validations for this API.

Parameters

See: Common Parameters, page 8-28

4. XLA_EVENTS_PUB_PKG.DELETE_BULK_EVENTS()

Summary

```
Procedure XLA_EVENTS_PUB_PKG.delete_bulk_events
(p_application_id           IN  INTEGER);
```

Description

This API deletes multiple events. Before calling this API, users must populate the XLA_EVENTS_INT_GT table with the following:

- application_id
- entity_code
- ledger_id
- event_id
- event_status_code

The API deletes events from the XLA_EVENTS table.

Validations

This API validates the application ID, event entity, and event ID. The status of the event to be deleted cannot be processed.

Parameters

See: Common Parameters, page 8-28

Common Parameters

The tables below provide details on the parameters common to many event APIs.

See: PL/SQL Data Types, page 8-33

Transaction Identifiers

The table below describes transaction identifier attributes.

Transaction Identifiers Attributes

Parameter Name	Type	Description
p_transaction_number	Varchar2(240)	Transaction number of the transaction that the events are based on. This is the user transaction identifier and serves as a reference for the transaction.
p_event_source_info	xla_events_pub_pkg.t_event_source_info	System transaction identifiers (transaction identifiers); for example, invoice ID

Contextual Information

The table below describes contextual information details.

Contextual Information Details

Parameter Name	Type	Description
p_source_application_id	Integer	Internal identifier of the application that generates the document or transaction. This may be different from the application that generates and/or owns the accounting for the corresponding event. Source applications do not need to be registered as subledger applications. For example, in Oracle Manufacturing, transactions are generated by applications such as Receiving or Work in Process, while the accounting is generated and owned by Oracle Cost Management. Therefore, p_source_application_id corresponds to the internal identifier of Receiving or Work in Process, while p_application_id corresponds to the internal identifier of Oracle Cost Management. If no value is provided, the default is p_application_id.
p_application_id	Integer	Application internal identifier
p_legal_entity_id	Integer	Legal entity internal identifier
p_ledger_id	Integer	Ledger internal identifier
p_valuation_method	Varchar2(30)	Valuation method used for securing a transaction. Some applications secure their transactions by valuation method. The method is stored as contextual data for events.

See:Create Accounting Concurrent Request, page 9-13

Transaction Security Identifiers

The table below describes transaction security identifier attributes.

Transaction Security Identifiers Attributes

Parameter Name	Type	Description
p_security_context	xla_events_pub_pkg.t_Security	Security context information for the transaction which has created the events

Transaction Reference Information

The table below describes transaction reference parameter attributes.

Transaction Reference Parameters Attributes

Parameter Name	Type	Description
p_array_reference_info	xla_events_pub_pkg.t_array_event_reference_info	Array of optional reference information for multiple events. These are stored with the events.
p_reference_info	xla_events_pub_pkg.t_event_reference_info	Optional reference information for a particular event

Event Information

The table below describes event information parameter attributes.

Event Information Parameters Attributes

Parameter Name	Type	Description
p_entity_type_code	Varchar2(30	Entity type internal code

Parameter Name	Type	Description
p_event_class_code	Varchar2(30)	Event class internal code
p_event_type_code	Varchar2(30)	Event type internal code
p_event_id	Integer	Event internal identifier
p_event_date	Date	Event GL date
p_event_status_code	Varchar2(1)	External status code for an event
p_event_number	Integer	Event sequence number within a transaction. Events are ordered by this sequence number for accounting.
p_array_entity_source_info	xla_events_pub_pkg.t_array_entity_source_info	Array of transaction source ID information as stamped on the Entity
p_array_event_type_code	xla_events_pub_pkg.t_array_event_type_code	Array of internal codes for the event type as defined by applications
p_array_event_date	xla_events_pub_pkg.t_array_event_date	Array of GL dates for events
p_array_event_status_code	xla_events_pub_pkg.t_array_event_status_code	Array of external status codes for events
p_array_event_number	xla_events_pub_pkg.t_array_event_number	Array of event sequence numbers within a transaction. Events are ordered by these sequence numbers for accounting.
p_array_entity_event_info	xla_events_pub_pkg.t_array_entity_event_info	Array of combined entity and event attributes

See: Creating Subledger Journal Entries from Accounting Events, page 2-1

PL/SQL Data Types

The following are the predefined PL/SQL data structures available in the XLA_EVENTS_PUB_PKG package.

PL/SQL Record Structures

Transaction Source Information

```
TYPE t_event_source_info IS RECORD
(source_application_id        PLS_INTEGER         DEFAULT NULL
,application_id               PLS_INTEGER
,legal_entity_id              PLS_INTEGER
,ledger_id                    PLS_INTEGER
,entity_type_code             VARCHAR2(30)
,transaction_number           VARCHAR2(240)
,source_id_int_1              PLS_INTEGER
,source_id_int_2              PLS_INTEGER
,source_id_int_3              PLS_INTEGER
,source_id_int_4              PLS_INTEGER
,source_id_char_1             VARCHAR2(30)
,source_id_char_2             VARCHAR2(30)
,source_id_char_3             VARCHAR2(30)
,source_id_char_4              VARCHAR2(30));
```

The table below provides descriptions on select attributes listed above.

Attribute Table #1

Attribute	Description
source_application_id	See description for p_source_application_id, page 8-30
application_id	Application transaction owner identifier
legal_entity_id	Transaction legal entity identifier
ledger_id	Transaction ledger identifier
entity_type_code	Entity code as defined by applications during setup
transaction_number	Transaction number of the transaction that has created the events. The transaction number serves as a reference for the transaction.

Attribute	Description
source_id_xxx_n	Generic columns that store the identifier for the transaction in the transaction table

Event Reference Information

```
TYPE t_event_reference_info IS RECORD
(reference_num_1                NUMBER
,reference_num_2                NUMBER
,reference_num_3                NUMBER
,reference_num_4                NUMBER
,reference_char_1               VARCHAR2(240)
,reference_char_2               VARCHAR2(240)
,reference_char_3               VARCHAR2(240)
,reference_char_4               VARCHAR2(240)
,reference_date_1               DATE
,reference_date_2               DATE
,reference_date_3               DATE
,reference_date_4               DATE);
```

See: Attribute Table #2, page 8-36

Event Information

```
TYPE t_event_info IS RECORD
(event_id              PLS_INTEGER
,event_number          NUMBER
,event_type_code       VARCHAR2(30)
,event_date            DATE
,event_status_code     VARCHAR2(1)
,process_status_code   VARCHAR2(1)
,reference_num_1       NUMBER
,reference_num_2       NUMBER
,reference_num_3       NUMBER
,reference_num_4       NUMBER
,reference_char_1      VARCHAR2(240)
,reference_char_2      VARCHAR2(240)
,reference_char_3      VARCHAR2(240)
,reference_char_4      VARCHAR2(240)
,reference_date_1      DATE
```

```
,reference_date_2      DATE
,reference_date_3      DATE
,reference_date_4      DATE);
```

See: Attribute Table #2, page 8-36

Entity and Event Information (including security contexts)

```
TYPE t_entity_event_info IS RECORD
(event_type_code              VARCHAR2(30)
,event_date                   DATE
,event_id                     PLS_INTEGER
,event_number                 NUMBER
,event_status_code            VARCHAR2(1)
,transaction_number           VARCHAR2(240)
,source_id_int_1              PLS_INTEGER
,source_id_int_2              PLS_INTEGER
,source_id_int_3              PLS_INTEGER
,source_id_int_4              PLS_INTEGER
,source_id_char_1             VARCHAR2(30)
,source_id_char_2             VARCHAR2(30)
,source_id_char_3             VARCHAR2(30)
,source_id_char_4             VARCHAR2(30)
,reference_num_1              NUMBER
,reference_num_2              NUMBER
,reference_num_3              NUMBER
,reference_num_4              NUMBER
,reference_char_1             VARCHAR2(240)
,reference_char_2             VARCHAR2(240)
,reference_char_3             VARCHAR2(240)
,reference_char_4             VARCHAR2(240)
,reference_date_1             DATE
,reference_date_2             DATE
,reference_date_3             DATE
,reference_date_4             DATE
,valuation_method             VARCHAR2(30)
,security_id_int_1            INTEGER
,security_id_int_2            INTEGER
,security_id_int_3            INTEGER
,security_id_char_1           VARCHAR2(30)
,security_id_char_2           VARCHAR2(30)
```

```
,security_id_char_3           VARCHAR2(30));
```

The table below provides descriptions on select attributes listed above.

Attribute Table #2

Attribute	Description
event_type_code	Code for the event type of the event, as defined during setup
event_date	Event GL date
event_id	Event internal identifier
event_number	Event sequence number for the event within a transaction. Events are processed in the order of their event number.
event_status_code	Status code for the event. This is the event's external status and is used by implementers.
transaction_number	Transaction number of the transaction that has created the events. The transaction number serves as a reference for the transaction.
source_id_xxx_n	Generic columns that store the identifier for the transaction in the transaction table
reference_xxx_n	Generic columns that store reference information for the event
valuation_method	Valuation method code used as a security context for applications that support the valuation method
security_id_xxx_n	Security contexts

Security Context Information

Security Context Information restricts the results of drill-downs. For example, setting the profile option SLA: Enable Subledger Transaction Security in GL to Yes for a specific user limits the transactions that a user can query.

See: Security Context Values, page 2-17

Use the following record structure to pass security context information through event

APIs. This structure is defined in XLA_EVENTS_PUB_PKG package.

```
TYPE t_security IS RECORD
(security_id_int_1   INTEGER
,security_id_int_2   INTEGER
,security_id_int_3   INTEGER
,security_id_char_1  VARCHAR2(30)
,security_id_char_2  VARCHAR2(30)
,security_id_char_3  VARCHAR2(30));
```

The table below provides descriptions for the attributes listed above.

Attribute Table #3

Attribute	Description
security_id_int_n	Security context information in INTEGER type
security_id_char_n	Security context information in VARCHAR type

PL/SQL Table Structures

Array of Information based on above Structures

```
TYPE t_array_event_reference_info IS  TABLE OF t_event_reference_info
TYPE t_array_event_info           IS  TABLE OF t_event_info
TYPE t_array_event_source_info    IS  TABLE OF t_event_source_info
TYPE t_array_entity_event_info    IS  TABLE OF t_entity_event_info
```

Other Array Structures

```
TYPE t_array_event_type           IS  TABLE OF VARCHAR2(30)
TYPE t_array_event_date           IS  TABLE OF DATE
TYPE t_array_event_status_code    IS  TABLE OF VARCHAR2(1)
TYPE t_array_entity_id            IS  TABLE OF PLS_INTEGER
TYPE t_array_event_id             IS  TABLE OF PLS_INTEGER
TYPE t_array_event_number         IS  TABLE OF NUMBER
```

Constants

The Financial Services Accounting Hub uses predefined accounting event status Constants. These constants are defined in the XLA_EVENTS_PUB_PKG package.

The table below lists event statuses and the corresponding constants that must be used

when employing event APIs.

Event Statuses and Constants

Event Status	Constant
Incomplete	XLA_EVENTS_PUB_PKG.C_EVENT_INCOMPLETE
Unprocessed	XLA_EVENTS_PUB_PKG.C_EVENT_UNPROCESSED
No Action	XLA_EVENTS_PUB_PKG.C_EVENT_NOACTION
Processed	XLA_EVENTS_PUB_PKG.C_EVENT_PROCESSED

9

Accounting Program and Period Close Integration

Accounting Program Workflow Business Events for Custom Integration

Note: Oracle Financial Services Accounting Hub provides workflow business events only to support customizations at the customer site. Therefore, they may not be used for integration between Oracle Applications and the Financial Services Accounting Hub.

These business events are seeded with the status Disabled. Customers need to enable them at their installations if they want to use them for customizations.

The Financial Services Accounting Hub enables customers to extend and customize the integration with the Financial Services Accounting Hub's Create Accounting program by providing business events they can subscribe to.

This chapter describes how to integrate with the Create Accounting program and the technical specifications on how to subscribe to the Accounting Program workflow business events. The workflow based business events architecture enables users to make Create Accounting program extensions more generic and extensible.

The Business Events System uses the Oracle Advanced Queuing (AQ) infrastructure to communicate business events between systems. The Business Events System consists of the Event Manager which enables users to do the following:

- Register subscriptions to events that are significant to their systems
- Model business events within workflow processes

When a local event occurs, the subscribing code is executed in the same transaction as the code that raised the event. Subscription processing can include the following:

- Executing custom code on the event information
- Sending event information to a workflow process
- Sending event information to other queues or systems

Currently, the Create Accounting program raises the following workflow business events described in the table below which shows the available business events in each mode.

Available Business Events in Document and Batch Mode

Business Events	Document Mode	Batch Mode
Pre-accounting		X
Transaction Objects	X	X
Post-processing	X	X
Post-accounting		X

Implementers of the Financial Services Accounting Hub at the customer site can subscribe to these workflow business events to perform the following application-specific tasks:

- Preaccounting
- Accounting program extract
- Postprocessing
- Postaccounting

> **Note:** This is only required if customers need to extend or customize the integration with the Accounting Program.

When the Create Accounting program is submitted, the preaccounting, extract, and postprocessing workflow business events are raised only if the Create Accounting flag is set to Yes. The postaccounting workflow business event is always raised no matter what the value of the Create Accounting flag is.

Technical Implementation Steps and Conventions

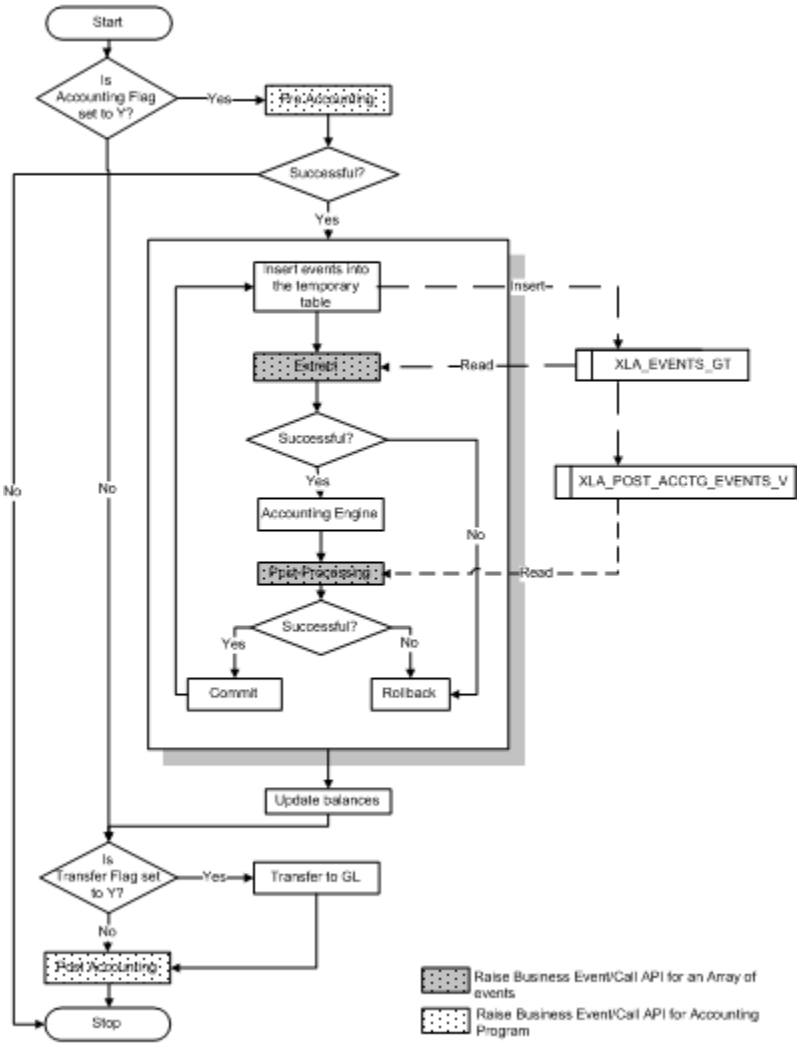
This section includes the following:

Timing and Position of Workflow Business Events

This section describes the life cycle of the Create Accounting program and the location of workflow business events.

The figure below shows the life cycle of the Create Accounting program and the location of workflow business events and is described in this chapter.

Create Accounting Process Life Cycle and Workflow Business Event:



The preaccounting event is raised at the beginning of the Create Accounting program after selecting accounting events for processing.

This section describes the following events:

- oracle.apps.xla.accounting.preaccounting, page 9-5
- oracle.apps.xla.accounting.extract (batch and document mode), page 9-5
- oracle.apps.xla.accounting.postprocessing (batch and document mode), page 9-5

- oracle.apps.xla.accounting.postaccounting (batch mode only), page 9-5

oracle.apps.xla.accounting.preaccounting (batch mode only)

The preaccounting event is raised at the beginning of the Create Accounting program after selecting accounting events for processing.

Example: Custom applications can use this business event to identify transactions for processing.

oracle.apps.xla.accounting.extract (batch and document mode)

This event is raised for all modes of accounting, final and draft, in batch as well as document mode. Events selected for the processing are passed to the subscription routine through the global temporary table XLA_EVENTS_GT. Implementers subscribing to this event can use it to populate transaction objects based on information in the XLA_EVENTS_GT table.

Example: Use this business event to populate transaction objects.

- Transaction Objects Introduction, page 3-1
- Seeding Calls to Subledger Event Capture Routines Introduction, page 8-1

oracle.apps.xla.accounting.postprocessing (batch and document mode)

The postprocessing event is raised after creating subledger journal entries for each processing unit (commit unit). This event is raised if the Create Accounting parameter is set to Yes. The subscription to the postprocessing event uses the view XLA_POST_ACCTG_EVENTS_V based on XLA_EVENTS_GT to derive accounting events successfully accounted by the Create Accounting program.

Example: Use this business event to update posted flags on the transaction distributions to indicate whether the transaction was accounted successfully.

oracle.apps.xla.accounting.postaccounting (batch mode only)

This event is raised after subledger journal entries are successfully committed in the database. If the Transfer to GL process is submitted as part of the accounting, then this workflow business event is raised after the transfer to General Ledger is completed successfully.

Example: A project is financially executed based on the funding that is set up according to the project's agreement. If funding is set up in foreign currency, the amount could change over time due to the fluctuation of the currency. In this example, the customer is maintaining the foreign currency amount in a separate custom table. Therefore, when the funding revaluation occurs, current funding is recalculated from the accounted amounts in the functional currency. In this case, the Funding Revaluation API uses the postaccounting business events.

The table below describes the view structure.

XLA_POST_ACCTG_EVENTS_V

Column Name	Null?	Data Type	Description
EVENT_ID	Null	NUMBER(15)	Event internal identifier
EVENT_NUMBER	Null	NUMBER(15)	Event number assigned to the event within the document
APPLICATION_ID	Not Null	NUMBER(15)	Internal identifier for the application to which the event belongs
EVENT_TYPE_CODE	Null	VARCHAR2(30)	Code for the event type that classifies the event being created
EVENT_DATE	Null	DATE	Event or accounting date for the event
ENTITY_ID	Null	NUMBER(15)	Internal identifier for the entity representing the actual document
EVENT_STATUS_CODE	Null	VARCHAR2(1)	Event status code
PROCESS_STATUS_CODE	Null	VARCHAR2(1)	Event process code
TRANSACTION_NUMBER	Null	VARCHAR2(240)	Transaction number given to the document by the products owning the document
LEDGER_ID	Not Null	NUMBER(15)	Ledger internal identifier to which event the belongs

Column Name	Null?	Data Type	Description
LEGAL_ENTITY_ID	Null	NUMBER(15)	Internal identifier for the legal entity
ENTITY_CODE	Not Null	VARCHAR2(30)	Event entity type code
SOURCE_ID_INT_1	Null	NUMBER(15)	Placeholder column that stores internal identifier of the document being represented by the entity
SOURCE_ID_INT_2	Null	NUMBER(15)	Placeholder column that stores internal identifier of the document being represented by the entity
SOURCE_ID_INT_3	Null	NUMBER(15)	Placeholder column that stores internal identifier of the document being represented by the entity
SOURCE_ID_INT_4	Null	NUMBER(15)	Placeholder column that stores internal identifier of the document being represented by the entity
SOURCE_ID_CHAR_1	Null	VARCHAR2(30)	Placeholder column that stores internal identifier of the document being represented by the entity

Column Name	Null?	Data Type	Description
SOURCE_ID_CHAR_2	Null	VARCHAR2(30)	Placeholder column that stores internal identifier of the document being represented by the entity
SOURCE_ID_CHAR_3	Null	VARCHAR2(30)	Placeholder column that stores internal identifier of the document being represented by the entity
SOURCE_ID_CHAR_4	Null	VARCHAR2(30)	Placeholder column that stores internal identifier of the document being represented by the entity
REFERENCE_NUM_1	Null	NUMBER	Reference information
REFERENCE_NUM_2	Null	NUMBER	Reference information
REFERENCE_NUM_3	Null	NUMBER	Reference information
REFERENCE_NUM_4	Null	NUMBER	Reference information
REFERENCE_CHAR_1	Null	VARCHAR2(240)	Reference information
REFERENCE_CHAR_2	Null	VARCHAR2(240)	Reference information
REFERENCE_CHAR_3	Null	VARCHAR2(240)	Reference information
REFERENCE_CHAR_4	Null	VARCHAR2(240)	Reference information
REFERENCE_DATE_1	Null	DATE	Reference information
REFERENCE_DATE_2	Null	DATE	Reference information
REFERENCE_DATE_3	Null	DATE	Reference information

Column Name	Null?	Data Type	Description
REFERENCE_DATE_4	Null	DATE	Reference information
EVENT_CREATED_BY	Null	VARCHAR2 (100)	Standard Who column

Parameter Specifications

This section describes parameters for the Create Accounting program workflow business events.

The table below describes the parameters for oracle.apps.xla.accounting.preaccounting.

oracle.apps.xla.accounting.preaccounting Parameters

Parameter Name	Description
APPLICATION_ID	Application identifier for which the Create Accounting program is submitted. The subscription routine checks this parameter first to see if it is the desired application.
LEDGER_ID	Ledger identifier for which the Create Accounting program is submitted
PROCESS_CATEGORY	Process category specified by users when launching the Create Accounting request
END_DATE	End date specified by users
ACCOUNTING_MODE	Indicates the mode in which the Create Accounting request is submitted
VALUATION_METHOD	Valuation method specified by users

Parameter Name	Description
SECURITY_ID_INT_1 SECURITY_ID_INT_2 SECURITY_ID_INT_3 SECURITY_ID_CHAR_1 SECURITY_ID_CHAR_2 SECURITY_ID_CHAR_3	Security context values as passed in as a parameter for the Create Accounting program

The table below describes the parameters for oracle.apps.xla.accounting.extract.

oracle.apps.xla.accounting.extract Parameters

Parameter Name	Description
APPLICATION_ID	Application identifier for which the Create Accounting program is submitted. The subscription checks this parameter first to see if it is the desired application.
ACCOUNTING_MODE	Indicates the mode in which the Create Accounting program is submitted

The table below describes the parameters for oracle.apps.xla.accounting.extract.

oracle.apps.xla.accounting.extract Parameters

Parameter Name	Description
APPLICATION_ID	Application identifier for which the accounting program is submitted. The subscription checks this parameter first to see if it is the desired application.
ACCOUNTING_MODE	Indicates the mode in which the Create Accounting program is submitted

oracle.apps.xla.accounting.postprocessing Parameters

These parameters are the same as the parameters defined for oracle.apps.xla.accounting.extract as described in the oracle.apps.xla.accounting.extract table, page 9-10 .

oracle.apps.xla.accounting.postaccounting Parameters

These parameters are the same as the parameters defined for oracle.apps.xla.accounting.preaccounting as described above in the oracle.apps.xla.accounting preaccounting table, page 9-9.

Application Subscription to Workflow Business Events

To integrate non-Oracle applications or customizations with the Financial Services Accounting Hub, you should subscribe to the workflow business event, if needed.

In the subscription routine, you can use the parameters passed with the workflow business event.

See: Parameter Specifications, page 9-9

When applying a subscription to a workflow business event, you should use a phase number less than 99 for an immediate or a synchronous execution. Those subscriptions with phase number less than 99 are executed in the same database session as the Create Accounting program. Uncommitted data created by the Create Accounting program is visible in the subscription routine. You can use a phase number greater than 100. In this case, the subscription routine is deferred, and it can only see the committed data.

In order to improve the performance, it is recommended that you check the application ID as the first step in the subscription and exit if it does not match.

Locking

Transaction data should not be changed when the document is being processed by the Create Accounting program. One way to achieve this is by marking the documents or distributions with an In Process status during the execution of the preaccounting business event. Another way to prevent documents from updating is to issue a row level lock on the documents during the execution of accounting transaction objects business event. In the second approach, the transaction objects procedure locks documents selected for the processing in the current processing unit in the NOWAIT mode to prevent any modifications to the document. The lock is released once the Create Accounting program issues the commit for the processing unit.

Transaction Control Statements

The workflow does not allow a commit in the subscription routine. If a subscription raises an exception, the Workflow Business Event Manager rolls back all the

subscriptions that were executed for that event and raises the exception to the Create Accounting program.

Exception Handling

In case of an error, the product routine should raise the standard application exception by calling APP_EXCEPTION.RAISE_EXCEPTION and stack the corresponding error message using FND_MESSAGE APIs. The Create Accounting Program handles this exception and either stores this message as Error for the event or prints this message in the concurrent log.

If the preaccounting subscription fails, the Create Accounting program exits without creating journal entries. If the postaccounting subscription fails, it does not affect journal entries created in that batch since the postaccounting workflow business event is raised after the changes are saved in the database.

If extract and postprocessing subscriptions fail, then the thread that encountered the exception rolls back the processing unit and comes out. Previously processed processing units remain committed. However, the other threads continue their processing.

Default Subscription to Postaccounting Workflow Business Event

The Financial Services Accounting Hub team has created a default subscription workflow to postaccounting workflow business event to notify the user of the status of the Create Accounting request.

Accounting Program Specifications

This section describes the Create Accounting program online and the batch mode API specifications.

Accounting Program Submission

The Create Accounting program generates subledger journal entries for a subledger application by processing accounting events.

See: Creating Subledger Journal Entries from Accounting Events, page 2-1

The Create Accounting program has the following execution modes:

- batch
- document
- budgetary control

In batch mode, the Create Accounting program creates journal entries for a batch of

documents or transactions. In this mode, the Create Accounting program is always submitted as a concurrent request.

In documentmode, subledger journal entries are created for events belonging to a single document or a transaction. In this mode, the Create Accounting program can be executed synchronously or asynchronously as a concurrent request.

In budgetary control mode, the Create Accounting program processes only those events from the events table that require budgetary control. When called in budgetary control mode, the funds availability API is called and the result is populated to the journal entries. If a line fails budgetary control for an encumbrance entry, the original amount is replaced by zero and the reserve amount for the encumbrance line is adjusted according.

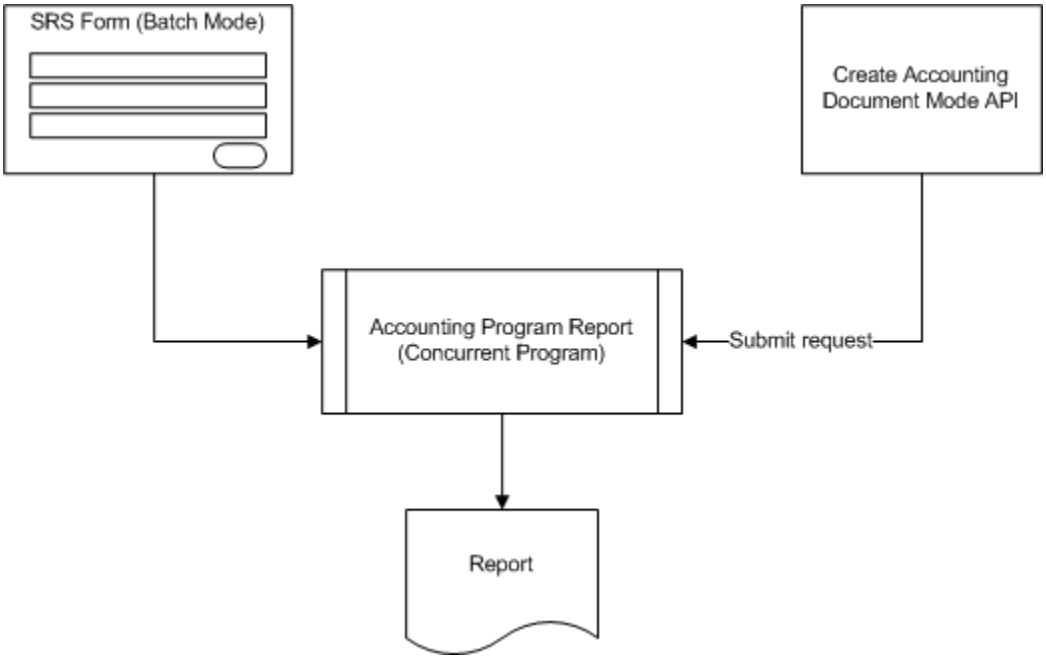
> **Note:** The budgetary control mode is only available through APIs and cannot be requested through a concurrent request submission.

Create Accounting Concurrent Request

The Create Accounting program executes the accounting program API for a single document or for a batch of documents. The program has input parameters that determine the execution mode and the selection criteria for events. Some of the parameters are specific to batchmode and some are specific to document mode.

The Create Accounting program is registered as a concurrent program. It can be submitted as a request from the standard request submission form for batch mode accounting. The figure below shows the create accounting submission process submitted by the concurrent program and the Create Accounting Document Mode API.

Create Accounting Program Submission



The table below describes the input parameters in Batch mode for the Create Accounting program.

Input Parameters in Batch Mode

Parameter Name	Description	Prompt	Standard Concurrent Program
P_ACCOUNTING_MODE	Accounting mode; D for draft, F for final	Mode	Displayed to users; value from a value set
P_APPLICATION_ID	Application identifier for which the Create Accounting program is going to execute		Not displayed to users; defaults from the profile option
P_CREATE_ACCOUNTING_FLAG	Indicates if the Create Accounting program is executed to create accounting or not; Y for yes, N for No	Create Accounting	Displayed to users. Value comes from a value set.

Parameter Name	Description	Prompt	Standard Concurrent Program
P_END_DATE	End date puts a filter on the selection of events. Only events having event date before the end date are selected for accounting.	End Date	Displayed to users; defaults to the SYSDATE
P_ERROR_ONLY_FLAG	Indicates if the accounting program should process only those entities that have failed events; Y for yes, N for no	Error Only	Displayed to users; value from a value set
P_GL_BATCH_NAME	Batch name used by Transfer to GL to decide on the batch name for the batch created in General Ledger	General Ledger Batch Name	Displayed to users; a free text field
P_LEDGER_ID	Internal identifier for the ledger for which the Create Accounting program is being executed	Ledger	Displayed to users; value from a value set
P_POST_IN_GL_FLAG	Indicates if users want to submit General Ledger posting; Y for yes, N for no	Post in General Ledger	Displayed to users; value from a value set
P_PROCESS_CATEGORY_CODE	Adds another filter criteria for the Create Accounting program to select events	Process Category	Displayed to users; value from a value set

Parameter Name	Description	Prompt	Standard Concurrent Program
P_REPORT_STYLE_FLAG	Users can choose to decide on the details of the report. The report can be printed in Summary (S) or Detail (D).	Report	Displayed to users; value from a value set
P_SECURITY_CHAR_1	Adds filter criteria for the event selection	N/A	N/A
P_SECURITY_CHAR_2	Adds filter criteria for the event selection	N/A	N/A
P_SECURITY_CHAR_3	Adds filter criteria for the event selection	N/A	N/A
P_SECURITY_INT_1	This adds filter criteria for the event selection.	N/A	N/A
P_SECURITY_INT_2	Adds filter criteria for the event selection	N/A	N/A
P_SECURITY_INT_3	Adds filter criteria for the event selection	N/A	N/A
P_TRANSFER_TO_GL_FLAG	Indicates whether the Create Accounting program should submit the Transfer to GL process	Transfer to General Ledger	Displayed to users; value from a value set
P_VALUATION_METHOD_CODE	Adds filter criteria for the event selection	N/A	N/A
P_SOURCE_APPLICATION_ID	Source application identifier. If specified, only accounting events created by this source application will be processed.		

See: Common Parameters, page 8-28

Accounting Program API (Document Mode)

The Financial Services Accounting Hub has defined two APIs to submit the Create Accounting program for a single document. These APIs can be called to execute the Create Accounting program synchronously or asynchronously as a concurrent request.

1.
```
PROCEDURE accounting_program_document
(p_event_source_info     IN xla_events_pub_pkg.t_event_source_info
,p_application ID     IN NUMBER   DEFAULT NULL
,p_entity_id             IN NUMBER
,p_accounting_flag       IN VARCHAR2  DEFAULT 'Y'
,p_accounting_mode       IN VARCHAR2
,p_transfer_flag         IN VARCHAR2
,p_gl_posting_flag       IN VARCHAR2
,p_offline_flag          IN VARCHAR2
,p_accounting_batch_id   OUT NOCOPY NUMBER
,p_errbuf                OUT NOCOPY VARCHAR2
,p_retcode               OUT NOCOPY NUMBER
,p_request_id            OUT NOCOPY NUMBER)IS
```

2.
```
PROCEDURE accounting_program_document
(p_event_source_info     IN xla_events_pub_pkg.t_event_source_info
,p_application_id      IN NUMBER  DEFAULT NULL
,p_valuation_method     IN VARCHAR2
,p_entity_id             IN NUMBER
,p_accounting_flag       IN VARCHAR2  DEFAULT 'Y'
,p_accounting_mode       IN VARCHAR2
,p_transfer_flag         IN VARCHAR2
,p_gl_posting_flag       IN VARCHAR2
,p_offline_flag          IN VARCHAR2
,p_accounting_batch_id   OUT NOCOPY NUMBER
,p_errbuf                OUT NOCOPY VARCHAR2
,p_retcode               OUT NOCOPY NUMBER
,p_request_id            OUT NOCOPY NUMBER)IS
```

The table below describes the parameters for the Accounting Program API, Document Mode.

Accounting Program API (Document Mode) Parameters

Parameter	Description
p_event_source_info	Record type gives the source information of the event and identifies the source transaction that needs accounting. If passed NULL, then a NOT NULL value should be passed in p_entity_id.
p_application_id	Application identifier for which the Create Accounting program is going to execute
p_entity_id	Internal identifier for the information represented by p_event_source_info. If a value is passed in this parameter, then the events of the document, represented by this ID, is accounted. If NULL is passed, then a valid value is expected in p_event_source_info.
p_accounting_flag	Indicates if the Create Accounting program should create accounting; Y for yes, N for no
p_accounting_mode	Indicates if the accounting being built is in Final (F) or Draft (D) mode
p_transfer_flag	Indicates if the Create Accounting program should transfer the entries to General Ledger; Y for yes, N for no
p_gl_posting_flag	Indicates if the GL Posting should be submitted for the entries created; Y for yes, N for no
p_offline_flag	Indicates whether the Create Accounting program should be submitted as a concurrent request (asynchronously) or should it be called as an API (synchronously); Y for asynchronous execution and N for synchronous execution
p_accounting_batch_id	OUT parameter; returns the accounting batch ID to the caller. The accounting batch ID is stamped on the journal entries created.

Parameter	Description
p_errbuf	OUT parameter; returns the completion message to the caller. The message can be an error message if the Create Accounting program encounters any fatal error.
p_retcode	This OUT parameter returns the success code back to the caller. • 0 indicates everything is normal and events are accounted successfully. • 1 indicates one or more events are accounted in error for the document. • 2 indicates there is some technical problem and the API has encountered an unexpected error. The API actions are rolled back.
p_request_id	If the Create Accounting program API is called with p_offline_flag = Y and it is submitted for asynchronous execution, this OUT parameter returns the request ID.
p_valuation_method	Valuation method context for the transaction that corresponds to p_entity_id

Accounting Program API (Batch Mode and Budgetary Control)

The Financial Services Accounting Hub has two APIs to create accounting online for a batch of documents or events called in batch mode. These APIs are also available in budgetary control mode.

1. XLA_ACCOUNTING_PUB_PKG.accounting_program_events (API)

   Use this API to create accounting for a list of events specified in the XLA_ACCT_PROG_EVENTS_GT table below. When this API is called in budgetary control mode, all events in the batch must be budgetary control enabled.

```
PROCEDURE accounting_program_doc_batch
(p_application id     IN INTEGER
,p_accounting_mode       IN VARCHAR2
,p_gl_posting_flag       IN VARCHAR2
,p_accounting_batch_id   IN OUT NOCOPY INTEGER
```

```
,p_errbuf                IN OUT NOCOPY VARCHAR2
,p_retcode               IN OUT NOCOPY INTEGER);
```

XLA_ACCT_PROG_EVENTS_GT

Column Name	Data Type	Size	Required
EVENT_ID	NUMBER	15	Yes

The table below describes the parameters for the XLA_ACCOUNTING_PUB_PKG.accounting_program_events API.

Parameter	Description
p_application_id	Application identifier
p_accounting_mode	Accounting mode. Valid values include: • DRAFT • FINAL • FUNDS_RESERVE • FUNDS_CHECK
p_accounting_batch_id	Accounting batch identifier
p_errbuf	Error buffer
p_retcode	Return code. Values include: • 0 - Process completed successfully. • 1 - Process completed with errors. • 2 - Exception occurred; process cannot be completed.

2. XLA_ACCOUNTING_PUB_PKG.accounting_program_doc_batch (API)

   Use this API to create accounting for a list of documents specified in the XLA_ACCT_PROG_DOCS_GT table below.

```
PROCEDURE accounting_program_events
(p_application id      IN INTEGER
,p_accounting_mode       IN VARCHAR2
,p_gl_posting_flag       IN VARCHAR2
,p_accounting_batch_id   IN OUT NOCOPY INTEGER
,p_errbuf                IN OUT NOCOPY VARCHAR2
,p_retcode               IN OUT NOCOPY INTEGER);
```

XLA_ACCT_PROG_DOCS_GT

Column Name	Data Type	Size
ENTITY_ID	NUMBER	15
LEDGER_ID	NUMBER	15
ENTITY_TYPE_CODE	VARCHAR2	30
VALUATION_METHOD	VARCHAR2	30
SOURCE_ID_INT_1	NUMBER	15
SOURCE_ID_INT_2	NUMBER	15
SOURCE_ID_INT_3	NUMBER	15
SOURCE_ID_INT_4	NUMBER	15
SOURCE_ID_CHAR_1	VARCHAR2	30
SOURCE_ID_CHAR_2	VARCHAR2	30
SOURCE_ID_CHAR_3	VARCHAR2	30
SOURCE_ID_CHAR_4	VARCHAR2	30

Each row in the XLA_ACCT_PROG_DOCS_GT represents one transaction.

If the entity_id is populated, it is used to identify the transaction. The entity_id must be a valid entity_id in the xla_transaction_entities table or an exception is raised.

If the entity_id is not populated, the rest of the columns are used to identify the

transaction. In this case, ledger_id and entity_type_code are required and the information should represent a valid entity in the xla_transaction_entities table. If the entity is not found, an exception is raised.

The table below describes the parameters for the Accounting Program API, Batch Mode.

Parameter	Description
p_application_id	Application identifier
p_accounting_mode	Accounting mode. Valid values include: • DRAFT • FINAL • FUNDS_RESERVE • FUNDS_CHECK
p_gl_posting_flag	Indicates if the GL Posting should be submitted for the entries created; Y for Yes, N for No
p_accounting_batch_id	Accounting batch identifier
p_errbuf	Error buffer
p_retcode	Return code. Values include: • 0 - Process completed successfully. • 1 - Process completed with errors. • 2 - Exception occurred; process cannot be completed.

Period Close Exceptions Report API

The XLA_EVENTS_PUB_PKG.PERIOD_CLOSE API is called by subledger application teams registered with the Financial Services Accounting Hub to check for any unaccounted events or untransferred journal entries for the application, ledger, and period.

See: Subledger Period Close Exceptions Report, *Oracle Subledger Accounting*

```
PROCEDURE XLA_EVENTS_PUB_PKG.PERIOD_CLOSE
(p_api_version
,x_return_status
,p_application_id
,p_ledger_id
,p_period_name);
```

Period Close Exceptions API Parameters Description

Parameter	Description	Data Type	Type	Required
p_api_version	Standard business object API parameter; compares the version number of incoming calls to its current version The current version is 1.0.	NUMBER	IN	YES

Parameter	Description	Data Type	Type	Required
x_return_status	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: • FND_API.G_RET_STS_SUCCESS: No period close exceptions • FND_API.G_EXC_ERROR: Period close exceptions exist • FND_API.G_EXC_UNEXPECTED_ERROR: Technical error	VARCHAR	OUT	N/A
p_application_id	Application identifier; must be a valid registered subledger application	INTEGER	IN	Yes
p_ledger_id	Subledger application ledger identifier	INTEGER	IN	Yes

Parameter	Description	Data Type	Type	Required
p_period_name	Period name derived from the accounting date for which the exceptions need to be checked	VARCHAR2	OUT	Yes

10

Gapless Event Creation and Processing

Gapless Event Creation and Processing Overview

Applications that generate messages in the process of transaction processing to capture appropriate accounting events use the gapless event creation and processing feature. Any number of distinct message brokers can handle or receive the messages generated. If one of the receiving units is delayed, the accounting event is captured out of order in the Financial Services Accounting Hub resulting in incomplete accounting.

If all the event numbers for a set of accounting events of an event entity transaction are consecutive starting from number 1 and there are no events with a status of Incomplete, then there is no gap. A gap exists if an event is missing or has a status of Incomplete. All events with an event number higher than the gap event are put on hold and cannot be processed until the gap is eliminated.

Example 1: An application generates three distinct types of accounting events for a given transaction: Transaction Created, Transaction Adjusted, and Transaction Canceled. The Transaction Canceled event is to be accounted as a reversal of the accounting generated for the Transaction Created and Transaction Adjusted events. For a given transaction, all three events are generated. However, due to some system delays, only the Transaction Created and Transaction Canceled events are created. The message containing the Transaction Adjusted event is not processed until after the Accounting Program runs and produces accounting for the Transaction Created and Transaction Canceled events. In this scenario, the accounting produced by the Transaction Canceled event does not include reversals for the entries generated by the Transaction Adjusted event.

Gapless event creation and processing allows implementers to enable gapless event processing for an event entity. For events created for an accounting event entity with gapless processing enabled, the Accounting Program only processes events for which all previous events for the same transaction have been captured and are either eligible for or have been processed. The Accounting Program does not handle dependencies across transactions or documents.

Example 2: An application with a transaction generates three distinct types of

accounting events: Transaction Created, Transaction Adjusted, and Transaction Canceled. The application marks the parent accounting event entity, Deals, as subject to gapless processing. The transaction generates in its life cycle five events as follows:

- One Transaction Created
- Three occurrences of Transaction Adjusted
- One Transaction Canceled

The implementer numbers the events as described in the table below.

Example 2: Numbered Events

Event Entity	Accounting Event Type	Event Date	Event Number
Deals	Transaction Created	10-OCT-2005	1
Deals	Transaction Adjusted	10-OCT-2005	2
Deals	Transaction Adjusted	11-OCT-2005	3
Deals	Transaction Adjusted	12-OCT-2005	4
Deals	Transaction Canceled	12-OCT-2005	5

Gapless Event Creation and Processing

This section includes the following:

Gapless Process Steps

Gapless event creation and processing includes the following steps:

1. The implementer enables gapless event processing in the Entities window.

   See: To Define Accounting Event Entities, page 4-12

2. The implementer assigns gapless numbers.

   Event entities are put on hold if there is a gap.

3. The Events API identifies gaps for accounting events belonging to an entity subject to gapless processing.

4. If the Events API identifies a gap before the newly created event, the new event is put on hold.

   In the Example 2: Numbered Events, page 10-2, if an event with event number 3 is missing or has a status of Incomplete, a gap is identified and events with event number 4 and 5 are put on hold.

5. Assuming that the event is not created with a status of Incomplete, when the creation of an event results in the elimination of a gap, all events belonging to the same transaction with higher event numbers are evaluated. For each of these events, if no other gaps remain, the event is released for processing.

Example 3: In Example 2: Numbered Events, page 10-2, assume that Event Numbers 2, 3, and 4 have not been created in the Financial Services Accounting Hub by the time Event Number 5 is created as shown in the table below.

Note: In this chapter, N indicates No and Y indicates Yes.

Example 3A

Event Entity	Accounting Event Type	Event Date	Event Number	Event Status	On Hold Flag
Deals	Transaction Created	10-OCT-2003	1	Unprocessed or Processed or No Action	N
Deals	Transaction Cancelled	12-OCT-2003	5	Unprocessed or No Action	Y

There are three gaps since events 2, 3, and 4 are missing.

If event 3 is created with a status of Unprocessed or No Action since event 2 is missing, event 3 is put on hold as shown in the table below.

Example 3B

Event Entity	Accounting Event Type	Event Date	Event Number	Event Status	On Hold Flag
Deals	Transaction Created	10-OCT-2003	1	Unprocessed or Processed or No Action	N
Deals	Transaction Adjusted	11-OCT-2003	3	Unprocessed or No Action	Y
Deals	Transaction Cancelled	12-OCT-2003	5	Unprocessed or No Action	Y

After event 3 is created, only gaps for events 2 and 4 remain. Assume that event number 2 is created. Since there is no gap before event 2, the hold flag for event 2 is set to N. Check whether a gap is eliminated by the creation of event 2. If event 2 is created with a status of Incomplete, then the gap due to event 2 is not eliminated, and it is not necessary to update the flag of the events with higher event numbers. If event 2 is created with another status, then the gap on event 2 is eliminated.

The next gap is on event 4, so the On Hold flag for event 3 (higher than 2, but lower than the next gap which is on event 4) is set to N if it is set to Y. This is described in Case 1 and Case 2 in the table below.

Example 3C, Case 1 and Case 2

Event Entity	Accounting Event Type	Event Date	Event Number	Case 1: Event Status	Case 1: On Hold Flag	Case 2: Event Status	Case 2: On Hold Flag
Deals	Transaction Created	10-OCT-2003	1	Unprocessed or Processed or No Action	N	Unprocessed or Processed or No Action	N
Deals	Transaction Adjusted	10-OCT-2003	2	Incomplete	N	Unprocessed or No Action	N

Event Entity	Accounting Event Type	Event Date	Event Number	Case 1: Event Status	Case 1: On Hold Flag	Case 2: Event Status	Case 2: On Hold Flag
Deals	Transaction Adjusted	11-OCT-2003	3	Unprocessed or No Action	Y	Unprocessed or No Action	N
Deals	Transaction Cancelled	12-OCT-2003	5	Unprocessed or No Action	Y	Unprocessed or No Action	Y

Event Updates and Deletes

The following gapless processing rules apply to event updates and deletes:

1. If a gap is eliminated by a change in the status of an event from Incomplete to Unprocessed or No Action or by a renumbering of the events for a transaction, then all events with a higher event number for which no other gap applies and which may have been placed on hold is released for processing.

2. If a gap is created by a change in status of an event from Unprocessed or No Action to Incomplete or by a renumbering of the events for a transaction, then all the events with a higher event number are put on hold.

3. Deletion of an event forces evaluation of all events with numbers higher than the deleted event. The On Hold flag of all affected events for which a gap was generated is updated to Y if it is set to N.

Example 4: The table below shows the state of accounting events for a transaction.

Example 4A

Event Entity	Accounting Event Type	Event Date	Event Number	Event Status	On Hold Flag
Deals	Transaction Created	10-OCT-2003	1	Unprocessed or Processed or No Action	N

Event Entity	Accounting Event Type	Event Date	Event Number	Event Status	On Hold Flag
Deals	Transaction Adjusted	10-OCT-2003	2	Unprocessed	N
Deals	Transaction Adjusted	11-OCT-2003	3	No Action	N
Deals	Transaction Adjusted	12-OCT-2003	4	No Action	N
Deals	Transaction Cancelled	12-OCT-2003	5	Unprocessed	N

The table below shows the results when an implementer requests the deletion of event number 2.

Example 4B

Event Entity	Accounting Event Type	Event Date	Event Number	Event Status	On Hold Flag
Deals	Transaction Created	10-OCT-2003	1	Unprocessed or Processed or No Action	N
Deals	Transaction Adjusted	11-OCT-2003	3	No Action	Y
Deals	Transaction Adjusted	12-OCT-2003	4	No Action	Y
Deals	Transaction Cancelled	12-OCT-2003	5	Unprocessed	Y

11

Financial Services Accounting Hub Security

Financial Services Accounting Hub Security Overview

Oracle Financial Services Accounting Hub uses accounting events generated by subledger applications to create accounting. As the implementation is generic for all applications, the architecture provides a common solution to enforce event security.

Subledger journal entries fall between the subledger applications and General Ledger. There are requirements to secure creation and inquiry of subledger journal entries and lines according to the following mechanisms:

- Transaction Security, page 11-2
- Flexfield Security Rules, page 11-6
- Data Access Set Security, page 11-7

The table below describes how the different security types are enforced in General Ledger and the Financial Services Accounting Hub.

Application	Transaction Security	Flexfield Security Rules	Transaction Security
General Ledger	Enforced by the Financial Services Accounting Hub inquiries (drilldown) and reports if the profile option SLA: Enabled Subledger Transaction Security in GL is set to Yes and the subledger application owning the transaction supports multiple operating units.	Always enforced by Financial Services Accounting Hub inquiries and reports.	Always enforced by Financial Services Accounting Hub inquiries and reports.
Subledger Application	Enforced by the Create Accounting program and the Financial Services Accounting Hub inquiries and reports if the subledger application provides a security policy.	Always enforced by manual journal entries and the Financial Services Accounting Hub inquiries and reports.	Enforced by the Create Accounting program, manual journal entries, and Financial Services Accounting Hub inquiries and reports if the profile option SLA: Enable Data Access Set Security in Subledgers is set to Yes.

Transaction Security

Transaction security refers to the way in which subledger transactions are secured. Subledger transactions are secured in different ways depending on the subledger application as in the following examples:

- In Payables and Receivables, the operating unit is used to secure transactions.
- Assets transactions are secured by asset book.
- Payroll transactions are secured by an HR specific security profile option.

Transaction Security Overview

The Financial Services Accounting Hub implements transaction security by securing the

accounting events.

The Financial Services Accounting Hub secures events by storing the security context of the transaction to which the event belongs. For example, if a transaction is secured by operating unit, events created for the transaction include operating unit data. Events therefore inherit the security of their source transactions.

Once event security is established for a subledger application, access from a subledger responsibility to the application's accounting events to view them, create accounting, or view the event's journal entries are limited by the security model of that application.

Example: Payables secures invoices by operating unit. For a customer using Payables, assume the following setup for transaction data that is described in the table below.

Example: Transaction Data Setup for a Customer Using Payables

Operating Unit	Invoice Number
North East	INV-101
North West	INV-102

A user without granted access to operating unit North West is not allowed to view accounting events, create accounting, or view accounting for INV-102. Similarly, a user with no access to operating unit North East is not allowed to view accounting events, create accounting, or view accounting for INV-101.

General Ledger responsibilities can view journal entries and balances in General Ledger independent of transaction security. However, drilldown through the Financial Services Accounting Hub optionally enforces transaction security. The profile option SLA: Enable Subledger Transaction Security in GL determines whether transaction security is applied.

- If the option is set to No for a General Ledger responsibility, inquiries and reports are not restricted by transaction security.
- If the option is set to Yes for a General Ledger responsibility, inquiries and reports are restricted by transaction security.

See: SLA: Enable Subledger Transaction Security in GL, *Oracle Subledger Accounting Implementation Guide*

Transaction Security Implementation

The Financial Services Accounting Hub supports transaction security through the standard security mechanisms supported by Oracle Applications.

When registering your application with the Financial Services Accounting Hub in the

Subledger Applications window, choose whether to use transaction security. If the subledger is set to enforce transaction security, then you are prompted to enter a policy function to be used in enforcing the security. This policy function must return the needed predicate (a string added to a WHERE clause in DML operations) that restricts the data that can be accessed on the secured objects.

If the subledger is not set to enforce transaction security, the Financial Services Accounting Hub automatically attaches a default policy that makes every rule accessible for the DML operations.

Information for events is implemented in the table. For every event that is created, the transaction security information, including the security context, is stored in the XLA_TRANSACTION_ENTITIES table. This table serves as a base object for event security in the XLA schema.

The table below lists the columns in the XLA_TRANSACTION_ENTITIES table that stores the security context for events.

XLA_TRANSACTIONS_ENTITIES Table

Col. Number	Security Context Columns	Data Type
1	Security_id_int_1	INTEGER
2	Security_id_int_2	INTEGER
3	security_id_int_3	INTEGER
4	security_id_char_1	VARCHAR2(30)
5	security_id_char_2	VARCHAR2(30)
6	security_id_char_3	VARCHAR2(30)
7	valuation_method	VARCHAR2(30)

To implement transaction security on accounting events, perform the following steps:

- Define a Policy Function, page 11-4
- Register Policy Functions in the Accounting Methods Builder, page 11-6

Define a Policy Function

As the security rules implied on a transaction are passed to the transaction's accounting events, the responsibility of defining security for events lie with the subledger

applications implementing the Financial Services Accounting Hub.

Security for the events in a subledger is defined through the use of a Policy Function. This function returns the needed predicate (a string added to a WHERE clause in DML operations) that can be used to secure the data in a table.

> **Note:** Functions should be written following the standards for application security policies given in Overview of Functions and Menu Security, *Oracle Applications Developer's Guide*.

Consider the following example of a policy function. Assume that Oracle Receivables implements the uptake of the Financial Services Accounting Hub and wants to implement transaction security on its accounting events. As Receivables secures its transactions using ORG_ID, it chooses to use the SECURITY_ID_INT_1 column to store its ORG_ID information when it creates its accounting events.

The security policy function could be as follows:

```
CREATE OR REPLACE PAKCAGE ar_security_pkg AS
FUNCTION ar_policy
(obj_schema VARCHAR2
,obj_name VARCHAR2) RETURN VARCHAR2
END ar_security_pkg;
/
CREATE OR REPLACE PACKAGE BODY ar_security_pkg as
FUNCTION ar_policy
(obj_schema VARCHAR2
,obj_name VARCHAR2) RETURN VARCHAR2 IS
BEGIN
RETURN 'EXISTS
(SELECT 1
FROM mo_glob_org_access_tmp oa
WHERE oa.organization_id =
xla_transaction_entities.security_id_int_1)';
END ar_policy;
END ar_security_pkg;
```

Whenever this policy function is active on the table, the function returns a predicate that is appended to all DML statements (Select, Update, and Delete) that are issued on the table XLA_TRANSACTION_ENTITIES.

In the example, a simple sql statement like:

```
SELECT *
from XLA_TRANSACTION_ENTITIES;
```

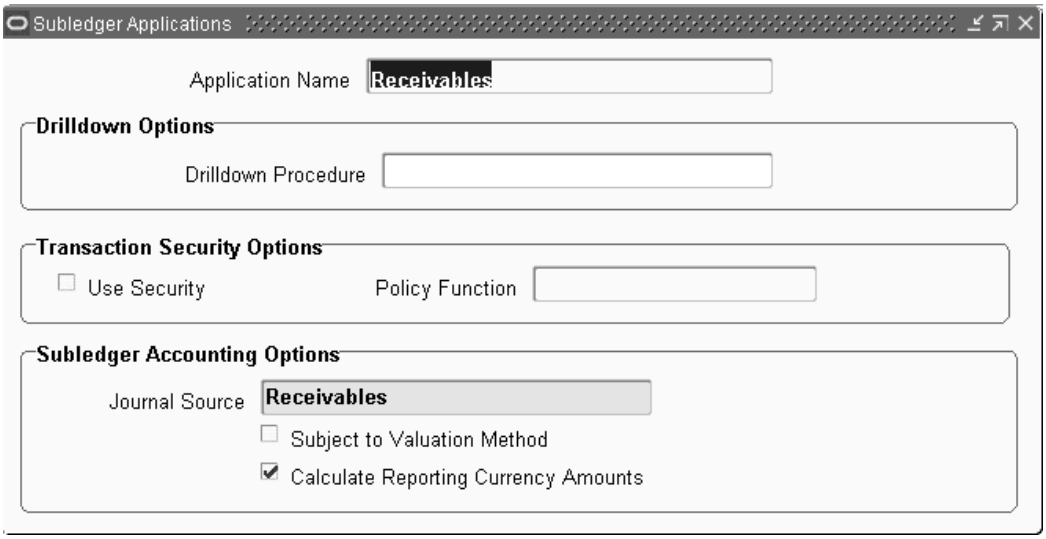
is now executed as:

```
SELECT *
from XLA_TRANSACTION_ENTITIES
WHERE EXISTS
(SELECT 1
FROM mo_glob_org_access_tmp oa
```

Register Policy Functions in the Accounting Methods Builder

Transaction security in the Financial Services Accounting Hub is implemented in the Transaction Security Options region of the Subledger Applications window.

See: Registering Subledger Applications, page 4-7



In the Event Security Options region of the Subledger Application window, select the appropriate option.

- If the application implements transaction security, select Use Security.
- If Use Security is selected, enter the name of the transaction in the Policy Function field.

  The Policy Function field is required if Use Security is selected. As the field has no validation, ensure that the value entered in this field corresponds to a valid function.

Flexfield Security Rules

Flexfield security rules are owned by the Application Object Library and enable users to restrict access to Accounting Flexfield segment values by responsibility. If users have access to the underlying subledger transactions, they should also have access to the Accounting Flexfield segments stored in the transaction data model. Therefore, the

Financial Services Accounting Hub does not validate whether the Accounting Flexfields stored in the transaction objects and used by the Accounting Program are consistent with flexfield security rules.

The following features implement flexfield security rules:

- Manual subledger journal entries
- Inquiries of subledger journal entry headers and lines
- Financial Services Accounting Hub reports

For Accounting Flexfield Security, no distinction is made between general ledger and subledger responsibilities in terms of security. Whether an inquiry is performed from a general ledger or a subledger responsibility, users cannot view journal entry lines containing code combinations unless they have access to all the segments in those combinations.

See: Overview of Flexfield Value Security, *Oracle Applications Flexfield Guide*

Data Access Set Security

Data access set security is enforced by Oracle General Ledger Data Access Sets. The Financial Services Accounting Hub makes data access set security optional for subledger responsibilities using the SLA: Enable Data Access Security in Subledgers profile option. The available values for this profile option are Yes and No. Data access set security is not optional for general ledger responsibilities.

Data access set security is available for the following:

- Journal entry creation (accounting program or manual subledger journal entries)
- Inquiries and reports of subledger journal entry headers and lines

See: SLA: Enable Data Access Security in Subledgers, *Oracle Subledger Accounting Implementation Guide*

Data Access Set

In addition to linking balancing and management segment values to ledgers, General Ledger grants access to ledgers and balancing or management segment values to user responsibilities using data access sets.

Data access set security refers to the data access set security features of the ledger architecture. A data access set grants read/write access to journal entries and lines by ledger, balancing segment values, or management segment values. A data access set can contain many ledgers that share the same chart of accounts and calendar. General Ledger supports one data access set per responsibility. Since primary and secondary representations of a subledger transaction typically have different charts of accounts,

subledger accounting supports up to two data access sets per responsibility. Assignment of data access sets to responsibilities is implemented using profile options. There are two profile options: GL: Data Access Set and SLA: Additional Data Access Set.

See:

- GL Profile Options, *Oracle General Ledger User Guide*
- SLA: Additional Data Access Set, *Oracle Subledger Accounting Implementation Guide*

Each data access set has the following levels:

- Ledgers: the ledgers to which the user's responsibility has access

  A data access set must have at least one ledger. All ledgers in a data access set must use the same chart of accounts and calendar.

- Balancing or management segment values

  Optionally, access can be granted to specific balancing segment or management segment values. The balancing and management segment level grants are read only or read/write.

Data Access Set Example

For the same enterprise, there could be two types of users:

- Users who have access to the Consulting Line of Business in all ledgers but not the Sales Line of Business
- Users who have access to all entries for the USA Corp. but not Canada Corp.

The table below describes the user types example.

Example User Types

Responsibility	Data Access Set Name	Ledger	Balancing Segment Values	Management Segment Values	Access Level
Consulting	Consulting Line of Business in both countries	USA Corp., Canada Corp.	(all)	Consulting LOB	Read/Write
USA Corp.	All USA Corp.	USA Corp.	(all)	(all)	Read/Write

Data Access Set Security for Journal Entry Creation

Journal Entry Creation includes the following security checks:

1. When users submit the Create Accounting request, the list of values for ledgers is restricted if the profile option SLA: Enable Data Access Security in Subledgers is set to Yes. The ledgers displayed in the list of values includes those that are assigned to the data access sets specified by the profile options GL: Data Access Set and SLA: Additional Data Access Set. Users must have both read and write access to the ledgers as part of the definition of the data access sets.

2. If ledger security is enabled, the Financial Services Accounting Hub validates that the user has write access to the balancing or management segment values assigned to any code combination for the subledger journal entry lines for the ledger of the subledger journal entry. Validation is done based on the type of access set.

Data Access Set Security for Inquiries and Reports

When inquiries or reports are accessed from a general ledger responsibility, data access set security is enforced. The user cannot view a journal entry line unless the user has read access to the ledger and the balancing or management segment values used in the line.

If ledger security is enabled using the SLA: Enable Data Access Security in Subledgers profile option when inquiries or reports are accessed from a subledger application responsibility, then the user cannot view a journal entry line unless the user has read access to the ledger and the balancing management segment values used in the line.

12

Manual Subledger Journal Entries API

Manual Subledger Journal Entries API Overview

The manual subledger journal entries APIs enable implementers to create, update, delete, complete, and reverse manual subledger journal entries. This API creates one subledger journal entry header or line at a time; therefore, it is not intended to create subledger journal entries in bulk.

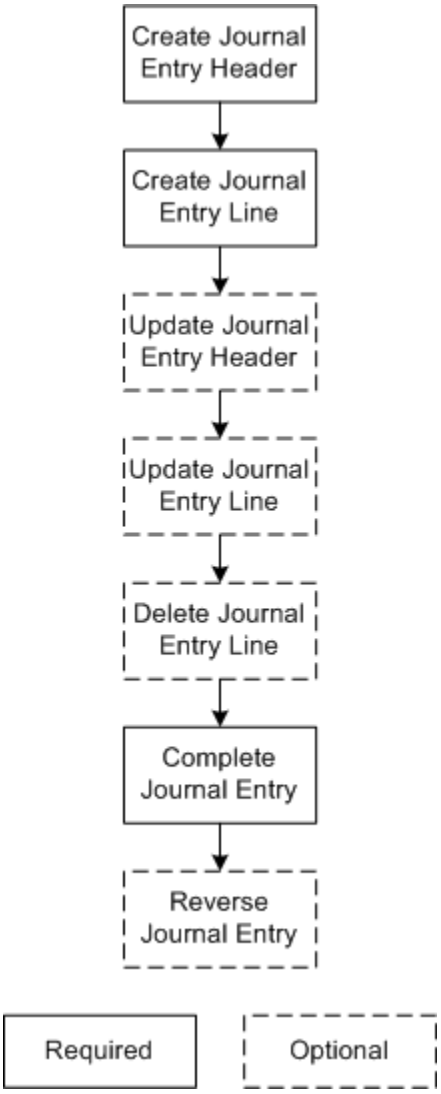
See: Subledger Journal Entry Definition Overview, *Oracle Subledger Accounting Implementation Guide*

Technical Overview

A manual subledger journal entry is not associated with a transaction or document of a subledger application. When users create a manual subledger journal entry, it is created with the event class and event type Manual that is seeded automatically when the subledger application is registered.

The figure below shows the create and complete manual journey entry process and is described in the following text.

Create and Complete Manual Journey Entry



The create and complete manual journal entry process includes the following steps:

1. Create journal entry header by calling CREATE_JOURNAL _ENTRY_HEADER API.

2. Create journal entry line by calling CREATE_JOURNAL_ENTRY_LINE API.

3. Optionally, update the journal entry header by calling UPDATE_JOURNAL_ENTRY_HEADER API.

4. Optionally, update the journal entry line by calling UPDATE_JOURNAL_ENTRY_LINE API.

5. Optionally, delete the journal entry line by calling DELETE_JOURNAL_ENTRY_LINE API.

6. Complete the journal entry by calling COMPLETE_JOURNAL_ENTRY API.

7. Optionally, reverse the journal entry by calling REVERSE_JOURNAL_ENTRY API.

Users can delete the journal entry at anytime by calling DELETE_JOURNAL_ENTRY API if the journal entry is not transferred to General Ledger.

XLA_JOURNAL_ENTRIES_PUB_PKG

This section includes the following topics:

Global Constants

The table below describes the constants that are defined in the package specifications.

Constants Defined in the Package Specifications

Name	Description	Date Type	Value
C_TYPE_MANUAL	Journal entry type: Manual	VARCHAR2	MANUAL
C_TYPE_UPGRADE	Journal entry type: Upgrade	VARCHAR2	UPGRADE
C_TYPE_MERGE	Journal entry type: Merge	VARCHAR2	MERGE
C_COMPLETION_OPTION_SAVE	Completion option: Save as Incomplete	VARCHAR2	S
C_COMPLETION_OPTION_DRAFT	Completion option: Draft	VARCHAR2	D
C_COMPLETION_OPTION_FINAL	Completion option: Final	VARCHAR2	F

Name	Description	Date Type	Value
C_COMPLATION_OPTION_TRANSFER	Completion option: Final and Transfer	VARCHAR2	T
C_COMPLETION_OPTION_POST	Completion option: Final, Transfer, and Post	VARCHAR2	P
C_REVERSAL_CHANGE_SIGN	Reversal option: Change Sign	VARCHAR2	SIGN
C_REVERSAL_SWITCH_DR_CR	Reversal option: Switch DR/CR	VARCHAR2	SIDE
C_STATUS_INCOMPLETE	Journal entry status: Incomplete	VARCHAR2	N
C_STATUS_INVALID	Journal entry status: Invalid	VARCHAR2	I
C_STATUS_DRAFT	Journal entry status: Draft	VARCHAR2	D
C_STATUS_FINAL	Journal entry status: Final	VARCHAR2	F
C_DELETE_NORMAL_MODE	Deletion mode: Normal	VARCHAR2	N
C_DELETE_FORCE_MODE	Deletion mode: Force	VARCHAR2	F

Published Journal Entries APIs

The following APIs are available in the XLA_JOURNAL_ENTRIES_PUB_PKG:

- CREATE_JOURNAL_ENTRY_HEADER, page 12-5
- UPDATE_JOURNAL_ENTRY_HEADER, page 12-12
- DELETE_JOURNAL_ENTRY, page 12-18
- CREATE_JOURNAL_ENTRY_LINE, page 12-21

- UPDATE_JOURNAL_ENTRY_LINE, page 12-29
- DELETE_JOURNAL_ENTRY_LINE, page 12-38
- COMPLETE_JOURNAL_ENTRY, page 12-42
- REVERSE_JOURNAL_ENTRY, page 12-47

CREATE_JOURNAL_ENTRY_HEADER

The CREATE_JOURNAL_ENTRY_HEADER API, as described in the table below, creates a new subledger journal entry header. The following conditions must be valid before the journal entry can be created:

- The P_APPLICATION_ID must be valid.
- The P_LEDGER_ID must be valid.
- The P_JE_CATEGORY_NAME must be valid.
- The P_BALANCE_TYPE_CODE must be valid.
- The P_GL_DATE must belong to a valid accounting period.

Any error found is imported in the message list which can be retrieved from the message list utility function.

The journal entry is created as Incomplete.

Once the entry is created, the output parameter X_AE_HEADER_ID represents the unique identifier of the entry created. It is used in other APIs to uniquely identify the journal entry.

CREATE_JOURNAL_ENTRY_HEADER Parameters

Parameter	Description	Date Type	Type	Required
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes

Parameter	Description	Date Type	Type	Required
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A

Parameter	Description	Date Type	Type	Required
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier; must be a valid application that is registered with the Financial Services Accounting Hub application	INTEGER	IN	Yes
P_LEDGER_ID	Ledger identifier	INTEGER	IN	Yes
P_LEGAL_ENTITY_ID	Legal entity identifier	INTEGER	IN	No
P_GL_DATE	GL date	DATE	IN	Yes
P_DESCRIPTION	Journal entry description	VARCHAR2	IN	Yes
P_JE_CATEGORY_NAME	Journal category name	VARCHAR2	IN	Yes

Parameter	Description	Date Type	Type	Required
P_BALANCE_TYPE_CODE	Balance type; must be one of the following values: A – Actual B – Budget E – Encumbrance	VARCHAR2	IN	Yes
P_BUDGET_VERSION_ID	Budget version identifier; required if BALANCE_TYPE_CODE is B	INTEGER	IN	No
P_ENCUMBRANCE_TYPE_ID	Encumbrance type identifier; required if BALANCE_TYPE_CODE is E	INTEGER	IN	No
P_REFERENCE_DATE	Journal entry reference date	DATE	IN	No
P_ATTRIBUTE_CATEGORY	Attribute category	VARCHAR2	IN	No
P_ATTRIBUTE1	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE2	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE3	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE4	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE5	Attribute column	VARCHAR2	IN	No

Parameter	Description	Date Type	Type	Required
P_ATTRIBUTE6	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE7	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE8	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE9	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE10	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE11	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE12	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE13	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE14	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE15	Attribute column	VARCHAR2	IN	No

Parameter	Description	Date Type	Type	Required
P_MSG_MODE	Mode of raising exception; must be one of the following: S – standard mode: The exception is added to the message list and can be retrieved using the message list utility function. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes
X_AE_HEADER_ID	Journal entry header identifier created by the API	INTEGER	OUT	N/A
X_EVENT_ID	Event internal identifier	INTEGER	OUT	N/A
X_PERIOD_NAME	Period name derived from accounting date	VARCHAR2	OUT	N/A
X_CREATION_DATE	Creation date	DATE	OUT	N/A
X_CREATED_BY	Created by identifier	INTEGER	OUT	N/A
X_LAST_UPDATE_DATE	Last update date	DATE	OUT	N/A

Parameter	Description	Date Type	Type	Required
X_LAST_UPDATED_BY	Last updated by identifier	INTEGER	OUT	N/A
X_LAST_UPDATE_LOGIN	Last update login identifier	INTEGER	OUT	N/A

UPDATE_JOURNAL_ENTRY_HEADER

The UPDATE_JOURNAL_ENTRY_HEADER API, described in the table below, modifies an existing manual subledger journal entry header. The following conditions must be valid before the journal entry can be updated:

- The journal entry is not Final.
- The P_JE_CATEGORY_NAME must be valid.
- The P_GL_DATE must belong to a valid accounting period.

Any error found is reported in the message list, which can be retrieved from the message list utility function.

If the status of the journal entry is draft, the draft balance calculation is undone and the draft journal entries created for reporting currencies is deleted.

Once the journal entry is updated, the status is updated to Complete.

The input parameters P_APPLICATION_ID and P_AE_HEADER_ID uniquely identify the journal entry to be updated. Other than the two parameters, all other input parameters are updated with the new values provided to the API.

UPDATE_JOURNAL_ENTRY_HEADER

Parameter	Description	Data Type	Type	Default
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes

Parameter	Description	Data Type	Type	Default
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A

Parameter	Description	Data Type	Type	Default
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier; must be a valid application that is registered with the Financial Services Accounting Hub application	INTEGER	IN	Yes
P_AE_HEADER_ID	Journal entry header identifier	INTEGER	IN	Yes
P_LEGAL_ENTITY_ID	Legal entity identifier Default is No.	INTEGER	IN	No
P_GL_DATE	GL date	DATE	IN	Yes
P_DESCRIPTION	Description	VARCHAR2	IN	Yes
P_JE_CATEGORY_NAME	Journal category name	VARCHAR2	IN	Yes
P_BUDGET_VERSION_ID	Balance type; required if it is a budget entry	INTEGER	IN	No

Parameter	Description	Data Type	Type	Default
P_ENCUMBRANCE_TYPE_ID	Encumbrance type; required if it is an encumbrance entry	INTEGER	IN	No
P_REFERENCE_DATE	Reference date	DATE	IN	No
P_ATTRIBUTE_CATEGORY	Attribute category	VARCHAR2	IN	No
P_ATTRIBUTE1	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE2	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE3	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE4	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE5	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE6	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE7	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE8	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE9	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE10	Attribute column	VARCHAR2	IN	No

Parameter	Description	Data Type	Type	Default
P_ATTRIBUTE11	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE12	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE13	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE14	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE15	Attribute column	VARCHAR2	IN	No
P_MSG_MODE	Mode of raising exception; must be one of the following: S – standard mode: The exception is added to the message list and can be retrieved using the message list utility function. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes
X_PERIOD_NAME	Period name derived from accounting date	VARCHAR2	OUT	N/A
X_LAST_UPDATE_DATE	Last update date	DATE	OUT	N/A

Parameter	Description	Data Type	Type	Default
X_LAST_UPDATED_BY	Last updated by identifier	INTEGER	OUT	N/A
X_LAST_UPDATE_LOGIN	Last update login identifier	INTEGER	OUT	N/A

DELETE_JOURNAL_ENTRY

The DELETE_JOURNAL_ENTRY API, described in the table below, deletes all information of a subledger journal entry including journal entry header and journal entry line. The subledger journal entry must not have been transferred or posted to General Ledger.

The input parameters P_APPLICATION_ID and P_AE_HEADER_ID uniquely identify the journal entry to be deleted.

When deleting a journal entry of status Final, the P_MODE must be F.

DELETE_JOURNAL_ENTRY

Parameter	Description	Data Type	Type	Default
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes

Parameter	Description	Data Type	Type	Default
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A

Parameter	Description	Data Type	Type	Default
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier	INTEGER	IN	Yes
P_AE_HEADER_ID	Journal entry header identifier	INTEGER	IN	Yes

Parameter	Description	Data Type	Type	Default
P_MODE	Deletion mode; must be one of the following values: N – Normal mode (default) F – Force mode; deletes all entries that are not posted	VARCHAR2	IN	Yes
P_MSG_MODE	Mode of raising exception; must be one of the following: S – standard mode: The exception is added to the message list and can be retrieved using the message list utility function. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes

CREATE_JOURNAL_ENTRY_LINE

The CREATE_JOURNAL_ENTRY_LINE API, described in the table below, creates a journal entry line for a subledger journal entry. The following conditions must be valid before the journal entry lines can be created for a journal entry:

- The P_APPLICATION_ID and P_AE_HEADER_ID must be valid.
- The journal entry is not Final.

- The P_DISPLAY_LINE_NUMBER must be unique for the journal entry.
- The P_CODE_COMBINATION_ID must be valid.

Any error found is reported in the message list, which can be retrieved from the message list utility function.

The input parameters P_APPLICATION_ID and P_AE_HEADER_ID uniquely identify the journal entry for which the journal entry line is created.

If the status of the journal entry is Draft, the draft balance calculation is undone and the draft journal entry created for reporting currency entries are deleted.

Once the journal entry line is created, the journal entry status is updated to Incomplete.

Once the journal entry line is created, the parameters P_APPLICATION_ID, P_AE_HEADER_ID, and X_AE_LINE_NUM represent the unique identifier of the journal entry line created. It is used in other APIs to uniquely identify the journal entry line.

CREATE_JOURNAL_ENTRY_LINE

Parameter	Description	Data Type	Type	Default
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes

Parameter	Description	Data Type	Type	Default
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A

Parameter	Description	Data Type	Type	Default
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier	INTEGER	IN	Yes
P_AE_HEADER_ID	Journal entry header identifier	INTEGER	IN	Yes
P_DISPLAYED_LINE_NUMBER	Displayed line number generated by the user	INTEGER	IN	Yes
P_CODE_COMBINATION_ID	Accounting code combination identifier	INTEGER	IN	Yes

Parameter	Description	Data Type	Type	Default
P_GL_TRANSFER_MODE	Must be one of the following values: D – Details S – Summary	VARCHAR2	IN	Yes
P_ACCOUNTING_CLASS_CODE	Accounting class code	VARCHAR2	IN	Yes
P_ENTERED_DR	Debit amount in transaction currency. Either X_ENTERED_DR or X_ENTERED_CR must be entered.	NUMBER	IN	No
P_ENTERED_CR	Credit amount in transaction currency. Either X_ENTERED_DR or X_ENTERED_CR must be entered.	NUMBER	IN	No
P_CURRENCY_CODE	Entered transaction currency. If the entered currency is different from the ledger currency, either accounted credit/debit or currency conversion information must be provided.	VARCHAR2	IN	No

Parameter	Description	Data Type	Type	Default
P_ACCOUNTED_DR	Debit amount in ledger currency	NUMBER	IN	No
P_ACCOUNTED_CR	Credit amount in ledger currency	NUMBER	IN	No
P_CONVERSION_TYPE	Conversion rate type for the foreign currency journal entry line	VARCHAR2	IN	No
P_CONVERSION_DATE	Date currency conversion rate is effective	DATE	IN	No
P_CONVERSION_RATE	Conversion rate for foreign currency; required if the conversion type is User	NUMBER	IN	No
P_PARTY_TYPE_CODE	Required if the natural account segment value of the Accounting Flexfield is defined as a control account. It must be one of the following values: C – Customer S – Supplier	VARCHAR2	IN	No
P_PARTY_ID	Third party for the journal entry line; required if PARTY_TYPE_CODE is entered; otherwise, this field is ignored.	NUMBER	IN	No

Parameter	Description	Data Type	Type	Default
P_PARTY_SITE_ID	Third party site for the journal entry line. This is an optional field, and the value is used only if PARTY_ID is not null.	NUMBER	IN	No
P_DESCRIPTION	Journal entry line description	VARCHAR2	IN	No
P_STATISTICAL_AMOUNT	Statistical amount	NUMBER	IN	No
P_JGZZ_RECON_FEF	Reconciliation reference to be imported into GL	VARCHAR2	IN	No
P_ATTRIBUTE_CATEGORY	Attribute category	VARCHAR2	IN	No
P_ATTRIBUTE1	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE2	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE3	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE4	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE5	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE6	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE7	Attribute column	VARCHAR2	IN	No

Parameter	Description	Data Type	Type	Default
P_ATTRIBUTE8	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE9	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE10	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE11	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE12	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE13	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE14	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE15	Attribute column	VARCHAR2	IN	No

Parameter	Description	Data Type	Type	Default
P_MSG_MODE	Mode of raising exception; must be one of the following: S – standard mode: The exception is added to the message list and can be retrieved using the message list utility function. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes
X_AE_LINE_NUM	Line number identifier	INTEGER	OUT	N/A
X_CREATION_DATE	Creation date	DATE	OUT	N/A
X_CREATED_BY	Created by identifier	INTEGER	OUT	N/A
X_LAST_UPDATE_DATE	Last update date	DATE	OUT	N/A
X_LAST_UPDATED_BY	Last updated by identifier	INTEGER	OUT	N/A
X_LAST_UPDATE_LOGIN	Last update login identifier	INTEGER	OUT	N/A

UPDATE_JOURNAL_ENTRY_LINE

The UPDATE_JOURNAL_ENTRY_LINE API, described in the table below, updates an

existing journal entry line. The following conditions must be valid before the journal entry line can be created for the journal entry:

- The P_APPLICATION_ID, P_AE_HEADER_ID, and P_AE_LINE_NUM must be valid.
- The journal entry is not Final.
- The P_DISPLAY_LINE_NUMBER must be unique for the journal entry.
- The P_CODE_COMBINATION_ID must be valid.

Any error found is reported in the message list, which can be retrieved from the message list utility function.

The input parameters P_APPLICATION_ID, P_AE_HEADER_ID, and P_AE_LINE_NUM uniquely identify the journal entry line to be updated. Other than these three parameters, all other input parameters are updated with the new values provided by the API.

If the status of the journal entry is Draft, the draft balance calculation is undone and the draft journal entry created for reporting currency entries are deleted.

UPDATE_JOURNAL_ENTRY_LINE

Parameter	Description	Date Type	Type	Default
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes

Parameter	Description	Date Type	Type	Default
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier	INTEGER	IN	Yes
P_AE_HEADER_ID	Journal entry header identifier	INTEGER	IN	Yes
P_AE_LINE_NUM	Journal entry line number	INTEGER	IN	Yes
P_DISPLAYED_LINE_NUMBER	Displayed line number used by the user	INTEGER	IN	Yes

Parameter	Description	Date Type	Type	Default
P_CODE_COMBINATION_ID	Accounting code combination identifier	INTEGER	IN	Yes
P_GL_TRANSFER_MODE	Must be one of the following values: D – Details S – Summary	VARCHAR2	IN	Yes
P_ACCOUNTING_CLASS_CODE	Accounting class code	VARCHAR2	IN	Yes
P_ENTERED_DR	Debit amount in transaction currency. Either X_ENTERED_DR or X_ENTERED_CR must be entered.	NUMBER	IN	No
P_ENTERED_CR	Credit amount in transaction currency. Either X_ENTERED_DR or X_ENTERED_CR must be entered.	NUMBER	IN	No

Parameter	Description	Date Type	Type	Default
P_CURRENCY_CODE	Entered currency of the transaction. If the entered currency is different from the ledger currency, either accounted credit/debit or currency conversion information must be provided.	VARCHAR2	IN	No
P_ACCOUNTED_DR	Debit amount in ledger currency	NUMBER	IN/OUT	
P_ACCOUNTED_CR	Credit amount in ledger currency	NUMBER	IN/OUT	
P_CONVERSION_TYPE	Exchange rate type for foreign currency journal entry line	VARCHAR2	IN	No
P_CONVERSION_DATE	Date currency conversion rate is effective	DATE	IN	No
P_CONVERSION_RATE	Exchange rate for foreign currency; required if the conversion type is User	NUMBER	IN	No

Parameter	Description	Date Type	Type	Default
P_PARTY_TYPE_CODE	Required if the natural account segment value of the Accounting Flexfield is defined as a control account. It must be one of the following values: C – Customer S – Supplier	VARCHAR2	IN	No
P_PARTY_ID	Third party for the journal entry line; required if PARTY_TYPE_CODE is entered. Otherwise, this field is ignored.	NUMBER	IN	No
P_PARTY_SITE_ID	Third party site for the journal entry line. This is an optional field, and the value is used only if PARTY_ID is not null.	NUMBER	IN	No
P_DESCRIPTION	Journal entry line description	VARCHAR2	IN	No
P_STATISTICAL_AMOUNT	Statistical amount	NUMBER	IN	No
P_JGZZ_RECON_FEF	Reconciliation reference to be imported into GL	VARCHAR2	IN	No

Parameter	Description	Date Type	Type	Default
P_ATTRIBUTE_CATEGORY	Attribute category	VARCHAR2	IN	No
P_ATTRIBUTE1	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE2	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE3	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE4	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE5	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE6	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE7	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE8	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE9	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE10	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE11	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE12	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE13	Attribute column	VARCHAR2	IN	No

Parameter	Description	Date Type	Type	Default
P_ATTRIBUTE14	Attribute column	VARCHAR2	IN	No
P_ATTRIBUTE15	Attribute column	VARCHAR2	IN	No
P_MSG_MODE	The mode of raising exception; must be one of the following: S – standard mode: The exception is added to the message list and can be retrieved using the message list utility function. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes
X_LAST_UPDATE_DATE	Last update date	DATE	OUT	N/A
X_LAST_UPDATED_BY	Last updated by identifier	INTEGER	OUT	N/A
X_LAST_UPDATE_LOGIN	Last update login identifier	INTEGER	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_RETCODE	Returns the status of the function, which is one of the following: 0 – completed successfully 1 – validation error found	INTEGER	OUT	N/A

DELETE_JOURNAL_ENTRY_LINE

The DELETE_JOURNAL_ENTRY_LINE API, described in the table below, deletes a subledger journal entry line. The following conditions must be valid before the journal entry line can be deleted for the journal entry:

- The P_APPLICATION_ID, P_AE_HEADER_ID, and P_AE_LINE_NUM must be valid.
- The journal entry is not Final.

Any error found is reported in the message list, which can be retrieved from the message list utility function.

The input parameters P_APPLICATION_ID, P_AE_HEADER_ID, and P_AE_LINE_NUM uniquely identify the journal entry line to be deleted.

Once the journal entry line is deleted, the journal entry status is updated to Incomplete.

DELETE_JOURNAL_ENTRY_LINE

Parameter	Description	Date Type	Type	Default
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes

Parameter	Description	Date Type	Type	Default
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier	INTEGER	IN	Yes
P_AE_HEADER_ID	Journal entry header identifier for which the journal entry line is deleted	INTEGER	IN	Yes
P_AE_LINE_NUM	Journal entry line number for which the journal entry line is deleted	INTEGER	IN	Yes
P_MSG_MODE	Mode of raising exception; must be one of the following: S – standard mode: The exception is raised immediately. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes

COMPLETE_JOURNAL_ENTRY

The COMPLETE_JOURNAL_ENTRY API, described in the table below, completes a subledger journal entry. The following conditions must be valid before the journal entry can be completed:

- The P_APPLICATION_ID and P_AE_HEADER_ID must be valid.
- The journal entry is not Final.

Any error found is reported in the message list, which can be retrieved from the message list utility function.

The input parameters P_APPLICATION_ID and P_AE_HEADER_ID uniquely identify the journal entry line to be deleted.

When completing the journal entry, validations are performed to ensure the journal entry is valid. If any validation error is found, the X_COMPLETION_STATUS returns X and the validation error can be found in the XLA_ACCOUNTING_ERRORS table.

COMPLETE_JOURNAL_ENTRY

Parameter	Description	Date Type	Type	Default
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes

Parameter	Description	Date Type	Type	Default
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier	INTEGER	IN	Yes
P_AE_HEADER_ID	Journal entry header identifier	INTEGER	IN	Yes

Parameter	Description	Date Type	Type	Default
P_COMPLETION_OPTION	The action to be performed to the journal entry; available options are as follows: D – complete draft accounting F – complete final accounting T – complete and transfer final accounting P – complete, transfer, and post final accounting	VARCHAR2	IN	Yes
P_MSG_MODE	The mode of raising exception; must be one of the following: S – standard mode: The exception is added to the message list and can be retrieved using the message list utility function. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes
X_AE_STATUS_CODE	Result journal entry status code	VARCHAR2	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_FUNDS_STATUS_CODE	Funds status code	VARCHAR2	OUT	N/A
X_COMPLETION_SEQ_VALUE	Completion sequence value	VARCHAR2	OUT	N/A
X_COMPLETION_SEQ_VER_ID	Completion sequence version id	INTEGER	OUT	N/A
X_COMPLETED_DATE	Completion date	DATE	OUT	N/A
X_GL_TRNSFER_STATUS_CODE	GL transfer status code	VARCHAR2	OUT	N/A
X_LAST_UPDATE_DATE	Last update date	DATE	OUT	N/A
X_LAST_UPDATED_BY	Last updated by identifier	INTEGER	OUT	N/A
X_LAST_UPDATE_LOGIN	Last update login identifier	INTEGER	OUT	N/A
X_TRANSFER_REQUEST_ID	Request identifier of the GL transfer request	INTEGER	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_RETCODE	Returns the status of the function, which is one of the following: C– completed successfully S– completed successfully and funds checker has reserved funds successfully A – completed successfully and funds checker has reserved funds with an Advisory status X – validation error is found and the journal entry cannot be completed F – funds reserve failed	VARCHAR2	OUT	N/A

REVERSE_JOURNAL_ENTRY

The REVERSE_JOURNAL_ENTRY API, described in the table below, reverses a subledger journal entry. The following conditions must be valid before the journal entry can be reversed.

- The P_APPLICATION_ID and P_AE_HEADER_ID must be valid.
- The journal entry must be Final.
- The P_REVERSAL_METHOD must be valid.
- The P_COMPLETION_OPTION must be valid.
- The P_GL_DATE must belong to a valid accounting period.

Any error found is reported in the message list which can be retrieved from the message list utility function.

The input parameters P_APPLICATION_ID and P_AE_HEADER_ID uniquely identify the journal entry line to be deleted.

When completing the journal entry, validations are performed to ensure the reversal journal entry is valid. If any validation error is found, the X_COMPLETION_STATUS returns X and the validation error can be found in the XLA_ACCOUNTING_ERRORS table.

REVERSE_JOURNAL_ENTRY

Parameter	Description	Date Type	Type	Default
P_API_VERSION	Standard business object API parameter; used by API to compare the version number of incoming calls to its current version number The current version of the API is 1.0	NUMBER	IN	Yes
P_INIT_MSG_LIST	Standard business object API parameter; allows the API callers to request that the API does the initialization of the message list on their behalf This parameter accepts the following values: FND_API_G_FALSE FND_API_G_FALSE_TRUE	VARCHAR2	IN	Yes

Parameter	Description	Date Type	Type	Default
X_RETURN_STATUS	Standard business object API parameter; represents the result of all the operations performed by the API and it has one of the following values: FND_API.G_RET_STS_SUCCESS FND_API.G_RET_STS_ERROR FND_API.G_RET_STS_UNEXP_ERROR	VARCHAR2	OUT	N/A
X_MSG_COUNT	Standard business object API parameter The APIs can generate none, one, or multiple messages and callers can call the API message list utility functions and procedures to retrieve those messages; holds the number of messages in the API message list.	NUMBER	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_MSG_DATA	Standard business object API parameter If the message count is one, this parameter holds the message in an encoded format.	VARCHAR2	OUT	N/A
P_APPLICATION_ID	Application identifier	INTEGER	IN	Yes
P_AE_HEADER_ID	Journal entry header identifier	INTEGER	IN	Yes
P_REVERSAL_METHOD	The reversal method; must be one of the following: SIGN – Change sign SIDE – Switch DR/CR	VARCHAR2	IN	Yes
P_GL_DATE	The GL date for the reversal journal entry	DATE	IN	Yes

Parameter	Description	Date Type	Type	Default
P_COMPLETION_OPTION	The action to be performed for the journal entry; available options are as follows: S – save as incomplete D – complete draft accounting F – complete final accounting T – complete and transfer final accounting P – complete, transfer, and post final accounting	VARCHAR2	IN	Yes
P_MSG_MODE	Mode of raising exception; must be one of the following: S – standard mode: The exception is raised immediately. O – OAF mode: The exception is added to the OA exception list and raised by calling raiseError() from the client side.	VARCHAR2	IN	Yes
X_REV_HEADER_ID	Journal entry identifier created for the reversal entry	INTEGER	OUT	N/A

Parameter	Description	Date Type	Type	Default
X_REV_EVENT_ID	Event identifier created for the reversal entry	INTEGER	OUT	N/A
X_COMPLETION_RETCODE	Returns the completion status of the function, which is one of the following: C – completed successfully S – completed successfully and funds checker has reversed funds successfully A – completed successfully and funds checker has reversed funds with an Advisory status X – validation error is found and the journal entry cannot be completed F – funds reserve failed	VARCHAR2	OUT	N/A
P_TRANSFER_REQUEST_ID	Request identifier of the GL transfer request	INTEGER	OUT	N/A

13

Drilldown

Drilldown Overview

This section includes the following parts:

- View Accounting, page 13-1
- Drilldown from General Ledger, page 13-2

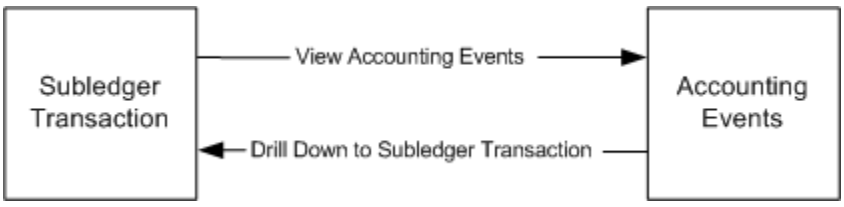
See:

- Subledger Accounting Events Inquiry Overview, *Oracle Subledger Accounting Implementation Guide*
- Subledger Journal Entry Lines Inquiry, *Oracle Subledger Accounting Implementation Guide*

View Accounting

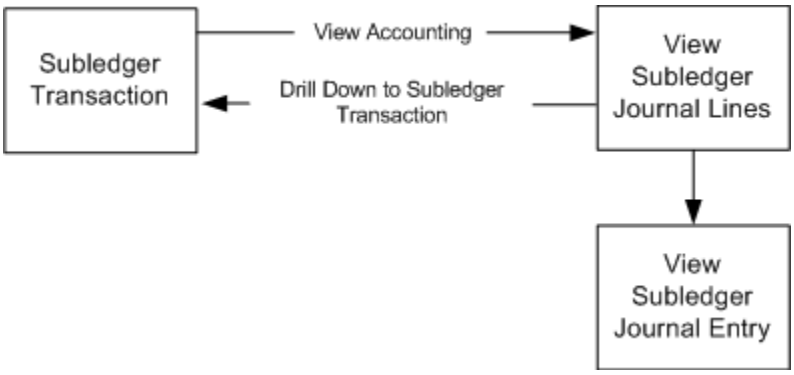
In each application that implements the Financial Services Accounting Hub, users can navigate to related subledger accounting events information and subledger journal entry information from a transaction.

The figure below shows the navigation from a subledger transaction to the accounting events or subledger journal entry lines. Users can drill down to the subledger transaction from the Accounting Events and the Subledger Journal Entry Lines pages.



From the Subledger Journal Entry Lines page, users can navigate to the subledger

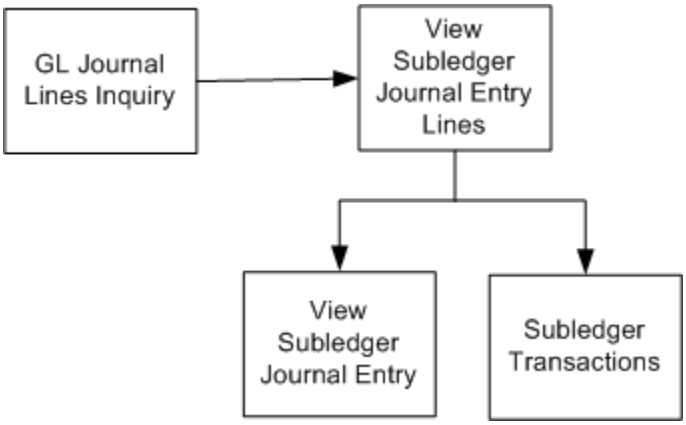
journal entry or drill down to the subledger transaction.



See: Drill-down API Details, page 13-2

Drilldown from General Ledger

The figure below shows the drilldown from GL journal lines inquiry to subledger journal entry lines in Subledger Accounting. From here, users can navigate to the subledger journal entries or drill down to the subledger transaction.



The drill-down procedure from the subledger journal entry to the subledger transaction must be registered with the Financial Services Accounting Hub.

See:

- Registering Subledger Applications, page 4-7
- Drill-down API Details, page 13-2

Drill-down API Details

The Financial Services Accounting Hub calls the Drill-down API to drill down as follows:

- From an accounting event or subledger journal entry to the subledger transaction
- From the subledger journal entry lines inquiry to the subledger transaction

  See: Subledger Journal Entry Lines Inquiry, *Oracle Subledger Accounting Implementation Guide*

The Drill-down API must be a PL/SQL procedure embedded in a package.

The Financial Services Accounting Hub recommends that you use the following naming convention for the drill-down procedure: <PRODUCT_SHORT_NAME>_DRILLDOWN_PUB_PKG.DRILLDOWN. For example, the procedure name for the Receivables drill-down procedure is: AR_DRILLDOWN_PUB_PKG.DRILLDOWN.

Based on the input, the API returns the appropriate information via OUT parameters to open the appropriate transaction form.

> **Note:** Although not all input parameters are applicable to your application, the procedure specification must contain all parameters listed in the API Specification section, page 13-3.

API Specification

```
<PRODUCT_SHORT_NAME>DRILLDOWN_PUB_PKG.DRILLDOWN
(
 p_application_id           IN              INTEGER
,p_ledger_id                IN              INTEGER
,p_legal_entity_id          IN              INTEGER    DEFAULT   NULL
,p_entity_code              IN              VARCHAR2
,p_event_class_code         IN              VARCHAR2
,p_event_type_code          IN              VARCHAR2
,p_source_id_int_1          IN              INTEGER    DEFAULT   NULL
,p_source_id_int_2          IN              INTEGER    DEFAULT   NULL
,p_source_id_int_3          IN              INTEGER    DEFAULT   NULL
,p_source_id_int_4          IN              INTEGER    DEFAULT   NULL
,p_source_id_char_1         IN              VARCHAR2   DEFAULT   NULL
,p_source_id_char_2         IN              VARCHAR2   DEFAULT   NULL
,p_source_id_char_3         IN              VARCHAR2   DEFAULT   NULL
,p_source_id_char_4         IN              VARCHAR2   DEFAULT   NULL
,p_security_id_int_1        IN              INTEGER    DEFAULT   NULL
,p_security_id_int_2        IN              INTEGER    DEFAULT   NULL
,p_security_id_int_3        IN              INTEGER    DEFAULT   NULL
,p_security_id_char_1       IN              VARCHAR2   DEFAULT   NULL
,p_security_id_char_2       IN              VARCHAR2   DEFAULT   NULL
,p_security_id_char_3       IN              VARCHAR2   DEFAULT   NULL
,p_valuation_method         IN              VARCHAR2   DEFAULT   NULL
,p_user_interface_type      IN  OUT  NOCOPY VARCHAR2
,p_function_name            IN  OUT  NOCOPY VARCHAR2
,p_parameters               IN  OUT  NOCOPY VARCHAR2

)
```

Parameter Descriptions

The table below describes the parameters.

Parameter Descriptions

Parameter	Description
p_application_id	Subledger application internal identifier
p_ledger_id	Event ledger identifier
p_legal_entity_id	Legal entity identifier
p_entity_code	Event entity internal code
p_event_class_code	Event class internal code
p_event_type_code	Event type internal code
p_source_id_int_1	Generic system transaction identifiers
p_source_id_int_2	Generic system transaction identifiers
p_source_id_int_3	Generic system transaction identifiers
p_source_id_int_4	Generic system transaction identifiers
p_source_id_char_1	Generic system transaction identifiers
p_source_id_char_2	Generic system transaction identifiers
p_source_id_char_3	Generic system transaction identifiers
p_source_id_char_4	Generic system transaction identifiers
p_security_id_int_1	Generic transaction security identifiers
p_security_id_int_2	Generic transaction security identifiers
p_security_id_int_3	Generic transaction security identifiers

Parameter	Description
p_security_id_char_1	Generic transaction security identifiers
p_security_id_char_2	Generic transaction security identifiers
p_security_id_char_3	Generic transaction security identifiers
p_valuation_method	Valuation Method internal identifier
p_user_interface_type	Determines the user interface type. Possible values: FORM - indicates that the source transaction is displayed using an Oracle*Forms based user interface HTML- indicates that the source transaction is displayed using HTML based user interface NONE- Use if the drill down is not supported for an event class or event type.
p_function_name	Name of the Oracle Application Object Library function defined to open the transaction form; used only if the page is a FORM page
p_parameters	An Oracle Application Object Library Function can have its own arguments or parameters. The Financial Services Accounting Hub expects implementers to return these arguments via p_parameters. This string can take any number of parameters and you can also use it to set some of the parameters dynamically. The additional parameters must be passed in the appropriate format. For the Oracle*Forms based UI the parameters must be space delimited (for example, "param1=value1 param2=value2"). For the HTML based UI, the parameters must be separated with & (for example, /OA_HTML/OA.jsp?OAFunc=function_name &param1=value1&parma2&value2).

See: Transaction Identifiers, page 2-7

Code Example

```
PACKAGE BODY AR_DRILLDOWN_PKG

IS

-- MODIFICATION HISTORY

-- Person      Date    Comments

-- ---------   ------  ------------------------------------------

-- Enter procedure, function bodies as shown below

PROCEDURE DRILLDOWN

(p_application_id            IN              INTEGER
,p_ledger_id                 IN              INTEGER
,p_legal_entity_id           IN              INTEGER    DEFAULT NULL
,p_entity_code               IN              VARCHAR2
,p_event_class_code          IN              VARCHAR2
,p_event_type_code           IN              VARCHAR2
,p_source_id_int_1           IN              INTEGER    DEFAULT NULL
,p_source_id_int_2           IN              INTEGER    DEFAULT NULL
,p_source_id_int_3           IN              INTEGER    DEFAULT NULL
,p_source_id_int_4           IN              INTEGER    DEFAULT NULL
,p_source_id_char_1          IN              VARCHAR2   DEFAULT NULL
,p_source_id_char_2          IN              VARCHAR2   DEFAULT NULL
,p_source_id_char_3          IN              VARCHAR2   DEFAULT NULL
,p_source_id_char_4          IN              VARCHAR2   DEFAULT NULL
,p_security_id_int_1         IN              INTEGER    DEFAULT NULL
,p_security_id_int_2         IN              INTEGER    DEFAULT NULL
,p_security_id_int_3         IN              INTEGER    DEFAULT NULL
,p_security_id_char_1        IN              VARCHAR2   DEFAULT NULL
,p_security_id_char_2        IN              VARCHAR2   DEFAULT NULL
,p_security_id_char_3        IN              VARCHAR2   DEFAULT NULL
,p_valuation_method          IN              VARCHAR2   DEFAULT NULL
,p_user_interface_type       IN  OUT  NOCOPY VARCHAR2
,p_function_name             IN  OUT  NOCOPY VARCHAR2
,p_parameters                IN  OUT  NOCOPY VARCHAR2   DEFAULT NULL )
IS

BEGIN

IF (p_application_id = 222) THEN

IF (p_event_class_code = 'INVOICE') THEN

p_user_interface_type := 'FORM'; -- Forms based UI

p_function_name      := 'XLA_ARXTWMAI';
p_parameters          := ' AR_TRANSACTION_ID="' ||TO_CHAR(p_source_id_
int_1)  ||'"'||' INVENTORY_ORG_ID="'||TO_CHAR(p_security_id_int_2)||'"'

||' ORG_ID="'||TO_CHAR(p_security_id_int_1)||'"';

ELSIF (p_event_class_code = 'RECEIPT') THEN

p_user_interface_type := 'FORM';

p_function_name       := 'XLA_ARXRWMAI';

p_parameters          := ' AR_RECEIPT_ID="' ||
TO_CHAR(p_source_id_int_1)||'"'||'ORG_ID="'||TO_CHAR(p_security_id_int_1
)||'"';
```

```
ELSIF (p_event_class_code = 'XXXX') THEN
p_user_interface_type := `HTML`; -- OA Framework Based UI
p_parameters          :=
'/OA_HTML/OA.jsp?OAFunc=function_name&param1=value1&parma2=value2';
ELSE
p_user_interface_type := 'NONE';
END IF;
END IF;
END DRILLDOWN;
END AR_DRILLDOWN_PKG;
```

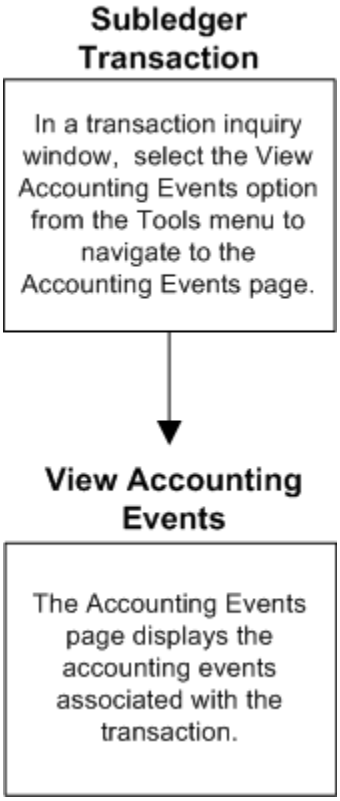
Drilldown Accounting from Transaction Workbench to Accounting Events

The Financial Services Accounting Hub recommends that implementers offer two drill-down options from transaction workbenches under the Tools menu option. The recommended options are as follows:

- View Accounting Events: the ability to inquire upon accounting events
- View Accounting: the ability to inquire upon subledger journal entry lines

From a transaction window, users can view accounting entry lines associated with that transaction in the Accounting Events page by selecting the View Accounting Events option from the Tools menu. Once the search region is populated, users can drill down to the subledger journal entry headers.The figure below shows this process.

Accounting Events Drilldown from the Transaction Workbench Process Flow



Specification

The function name used to call the Accounting Events engine is XLA_EVENT_FROM_TXN.

The table below describes the parameters for XLA_EVENT_FROM_TXN.

XLA_EVENT_FROM_TXN Parameters

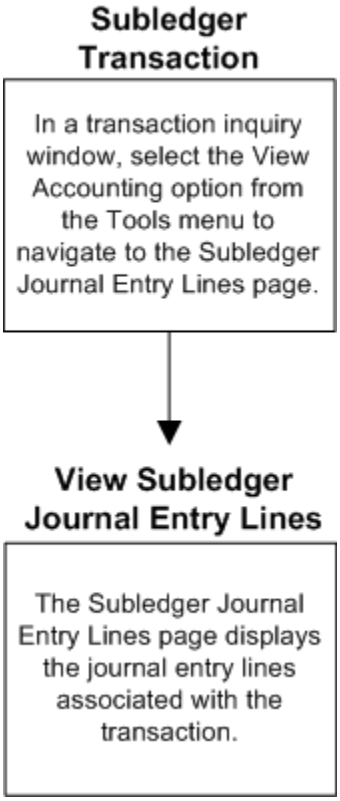
Parameter	Description
entityCode	Required; event entity internal code
sourceIdInt1	Generic system transaction identifiers
sourceIdInt2	Generic system transaction identifiers
sourceIdInt3	Generic system transaction identifiers

Parameter	Description
sourceIdInt4	Generic system transaction identifiers
sourceIdChar1	Generic system transaction identifiers
sourceIdChar2	Generic system transaction identifiers
sourceIdChar3	Generic system transaction identifiers
sourceIdChar4	Generic system transaction identifiers
returnToLabel	Optional; used only if the View Accounting page is called from an OA page and the implementers want to create a return to link in the View Accounting page to the calling page. You can use this parameter to indicate the label of the return to link. You can use an AOL message name in this parameter and specify the message application ID in the returnToApps parameter.
returnToApps	Optional; required only if returnToLabel is used and the AOL message name is used in the returnToLabel. It specifies the application of the AOL message name.
returnToLink	Optional; required if returnToLabel is specified. It is the URL of the return to link.

View Accounting from Transaction Workbench to Subledger Journal Entry Lines

From a transaction window, users can view subledger journal entry lines associated with that transaction in the View Subledger Journal Entry lines page by selecting the View Accounting option from the Tools menu. The figure below shows this process.

View Accounting Journal Entries for a Document or Transaction



Specification

The function name for calling the subledger journal lines engine is XLA_LINESINQ_SUBLEDGER.

The table below describes the parameters for XLA_LINESINQ_SUBLEDGER.

XLA_LINESINQ_SUBLEDGER Parameters

Parameter	Description
entityCode	Required; event entity internal code

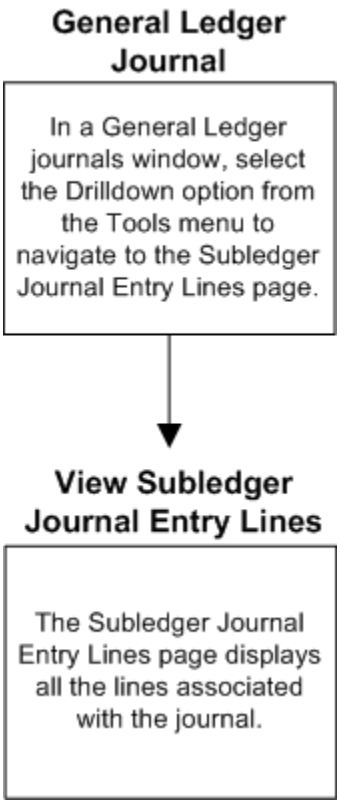
Parameter	Description
balanceTypeCode	Optional; balance type of the accountings to be retrieved. Values include: A – Actual B – Budget E – Encumbrance
sourceIdInt1	Generic system transaction identifiers
sourceIdInt2	Generic system transaction identifiers
sourceIdInt3	Generic system transaction identifiers
sourceIdInt4	Generic system transaction identifiers
sourceIdChar1	Generic system transaction identifiers
sourceIdChar2	Generic system transaction identifiers
sourceIdChar3	Generic system transaction identifiers
sourceIdChar4	Generic system transaction identifiers
returnToLabel	Optional; used only if the View Accounting page is called from an OA page and you want to create a return to link in the View Accounting page to the calling page. You can use this parameter to indicate the label of the return to link. You can use an AOL message name in this parameter and specify the message application ID in the returnToApps parameter.
returnToApps	Optional; required only if returnToLabel is used and AOL message name is used in the returnToLabel. It specifies the application of the AOL message name.
returnToLink	Optional; required if returnToLabel is specified. It is the URL of the return to link.

Parameter	Description
trxAppId	Optional; used in place of the session's application ID to query and view subledger journal entry lines created by another subledger application

Drilldown from General Ledger to Subledger Journal Entry Lines

From a General Ledger journal, users can view subledger journal entry lines associated with that journal in the View Subledger Journal Entry Lines page by selecting the Drilldown option from the Tools menu. The figure below shows this process.

Drilldown from General Ledger to Subledger Journal Entry Lines



Note: Ensure that the Import Journal References option is selected for the journal source in the General Ledger Journal Sources. If this option is not selected, General Ledger does not populate the GL_IMPORT_REFERENCES table. This table must be populated for the

drilldown to work.

14

Financial Services Accounting Hub Repository Specification

Financial Services Accounting Hub Repository Data Model

The Financial Services Accounting Hub repository is composed of the following entities:

- XLA_TRANSACTION_ENTITIES
- XLA_EVENTS
- XLA_AE_HEADERS
- XLA_AE_LINES
- XLA_DISTRIBUTION_LINKS

For information on these entities, see the Oracle Subledger Accounting eTRM.

Accessing the Financial Services Accounting Hub Repository

This section describes various ways to access the Financial Services Accounting Hub repository. All examples in this section are based on the Payables subledger. The examples are as follows:

Retrieve Journal Entries for a Transaction

Use the following SQL statements to retrieve all subledger journal entries for a transaction. Note that queries below do not return events in error or events with no journal entries.

Users can further restrict the query results by ledger (aeh.ledger_id), balance type (aeh.balance_type_code), and several other attributes available in the XLA_AE_HEADERS and XLA_AE_LINES tables.

SQL Statement – Retrieve all Journal Entries for a Transaction

The following statement returns all subledger journal entries created for the transaction.

```
select inv.invoice_num,
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
xla_transaction_entities ent,
xla_ae_headers           aeh,
xla_ae_lines             ael
where ent.application_id = 200
and inv.invoice_id     = ent.source_id_int_1
and ent.entity_code    = 'AP_INVOICES'
and ent.entity_id      = aeh.entity_id
and aeh.ae_header_id   = ael.ae_header_id
and inv.invoice_id     = 1234
```

SQL Statement – Retrieve Draft Journal Entries for a Transaction

Use the following SQL statement to retrieve draft journal entries for a transaction.

```
select inv.invoice_num,
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
xla_transaction_entities ent,
xla_ae_headers           aeh,
xla_ae_lines             ael
where ent.application_id = 200
```

```
and inv.invoice_id     = ent.source_id_int_1
and inv.invoice_id     = 1234
and ent.entity_code    = 'AP_INVOICES'
and ent.entity_id      = aeh.entity_id
and aeh.ae_header_id   = ael.ae_header_id
and aeh.accounting_entry_status_code = 'D'
```

SQL Statement – Retrieve Invalid Journal Entries for a Transaction

Use the following SQL statement to retrieve invalid journal entries for a transaction.

```
select inv.invoice_num,
aeh.ledger_id,
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
xla_transaction_entities ent,
xla_events               evt,
xla_ae_headers           aeh,
xla_ae_lines             ael
where ent.application_id = 200
and inv.invoice_id     = ent.source_id_int_1
and ent.entity_code    = 'AP_INVOICES'
and ent.entity_id      = evt.entity_id
and evt.event_id       = aeh.event_id
and aeh.ae_header_id   = ael.ae_header_id
and inv.invoice_id     = 1234
and evt.process_status_code = 'I'
```

Retrieve Journal Entries for an Event

Use the following SQL statements to retrieve subledger journal entries based on accounting events.

SQL Statement – Retrieve Journal Entries for a Single Event

Use the following SQL statement to retrieve journal entries for a specific event. It is assumed that users already know the event identifier. If users do not have the event identifier, it can be retrieved by using event APIs.

See: Overview of Event APIs, page 8-4

The SQL below retrieves subledger journal entries for the event id 201.

```
select inv.invoice_num,
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
xla_transaction_entities ent,
xla_events               evt,
xla_ae_headers           aeh,
xla_ae_lines             ael
where ent.application_id = 200
and inv.invoice_id     = ent.source_id_int_1
and ent.entity_code    = 'AP_INVOICES'
and ent.entity_id      = evt.entity_id
and evt.event_id       = aeh.event_id
and aeh.ae_header_id   = ael.ae_header_id
and inv.invoice_id     = 1234
and evt.event_id       = 201
```

SQL Statement – Retrieve Entries for Invalid Events

Use the following SQL statement to retrieve journal entries for invalid events.

```
select inv.invoice_num,
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
xla_transaction_entities ent,
xla_events               evt,
xla_ae_headers           aeh,
xla_ae_lines             ael
where ent.application_id = 200
and inv.invoice_id     = ent.source_id_int_1
and ent.entity_code    = 'AP_INVOICES'
and ent.entity_id      = evt.entity_id
and evt.event_id       = aeh.event_id
and aeh.ae_header_id   = ael.ae_header_id
```

```
and evt.process_status_code = 'I'
```

Retrieve Journal Entries for a Ledger

SQL Statement – Retrieve journal entries for the primary ledger

Assuming that the primary ledger is stored on the transaction, use the following SQL statement to retrieve journal entries for a primary ledger. The assumption here is that the inv.set_of_books_id contains the primary ledger identifier.

```
select inv.invoice_num,
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
xla_transaction_entities ent,
xla_events               evt,
xla_ae_headers           aeh,
xla_ae_lines             ael
where ent.application_id = 200
and inv.invoice_id     = ent.source_id_int_1
and ent.entity_code    = 'AP_INVOICES'
and ent.entity_id      = evt.entity_id
and evt.event_id       = aeh.event_id
and aeh.ae_header_id   = ael.ae_header_id
and aeh.ledger_id      = inv.set_of_books_id
```

SQL Statement – Retrieve Journal Entries for a Secondary Ledger

Use the following query to retrieve journal entries for a transaction and a secondary ledger.

Users must retrieve the secondary ledger identifier from the accounting setup before using this query. Users can use the same query to retrieve journal entries for reporting currencies.

```
select inv.invoice_num,
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
xla_transaction_entities ent,
```

```
xla_ae_headers           aeh,
xla_ae_lines             ael
where ent.application_id = 200
and inv.invoice_id     = ent.source_id_int_1
and ent.entity_code    = 'AP_INVOICES'
and ent.entity_id      = aeh.entity_id
and aeh.ae_header_id   = ael.ae_header_id
and aeh.ledger_id      = <secondary_ledger_id>
```

Retrieve Journal Entries Using Distribution Links

The distribution links table maintains a link between the subledger transaction distributions and subledger journal entry lines. Users can use this information for audit purposes or to inquire upon subledger journal lines related to a particular distribution.

SQL Statement – Retrieve Journal Entry Lines Related to Distributions

Use the following SQL statement to retrieve journal entries lines related to transaction distributions.

```
select inv.invoice_num,
dis.invoice_distribution_id
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
ap_invoice_distributions_dis_all,
xla_distribution_links   lnk,
xla_ae_headers           aeh,
xla_ae_lines             ael
where inv.invoice_id                = dis.invoice_id
AND inv.invoice_id                = 1234
AND lnk.application_id            = 200
AND lnk.SOURCE_DISTRIBUTION_TYPE  = 'AP_INVOICE_DISTRIBUTIONS'
AND dis.invoice_distribution_id   = lnk.source_distribution_id_num_1
AND lnk.ae_header_id              = ael.ae_header_id
AND lnk.ae_line_num               = ael.ae_line_num
AND aeh.ae_header_id              = ael.ae_header_id
```

SQL Statement – Retrieve Journal Entry Lines Related to Distributions/Event

Use the following SQL statement to retrieve journal entries lines related to transaction

distributions if the application stores the event identifier on the distribution.

```
select inv.invoice_num,
dis.invoice_distribution_id
ael.entered_dr,
ael.entered_cr,
ael.accounted_dr,
ael.accounted_cr
from ap_invoices_all          inv,
ap_invoice_distributions_dis_all,
xla_distribution_links   lnk,
xla_ae_headers           aeh,
xla_ae_lines             ael
where inv.invoice_id                = dis.invoice_id
AND inv.invoice_id                = 1234
AND lnk.application_id            = 200
AND dis.accounting_event_id       = lnk.event_id

AND lnk.ae_header_id              = ael.ae_header_id
AND lnk.ae_line_num               = ael.ae_line_num
AND aeh.ae_header_id              = ael.ae_header_id
```

15

Calculation of Accounted and Gain or Loss Amounts

Calculation of Accounted and Gain or Loss Amounts Overview

Oracle Financial Services Accounting Hub provides for the calculation of accounted and gain or loss amounts in a common framework across applications.

The calculation of accounted and gain or loss amounts feature provides the following:

- A consistent method to round amounts to the precision or minimum accountable unit of the currency and by supporting all valid conversion rate types defined in General Ledger
- The ability to calculate gain or loss amounts and account for it based on configuration defined by users in the Journal Line Types window
- The ability to enable or disable the automated calculation of accounted amounts and/or calculation of gain or loss amounts based on event class

Calculation of Accounted Amounts

If the Calculate Accounted Amounts option is enabled for the event class, Financial Services Accounting Hub uses the conversion information provided in the transaction objects to calculate accounted amounts, which is the amount in the functional currency of the ledger. When Calculate Accounted Amount is enabled, Calculate Gain or Loss Amounts must also be enabled.

It is recommended that unrounded entered amounts be provided in the transaction object to ensure a more accurate calculation of the accounted amount, gain or loss amount, and rounding difference.

Accounted amounts are calculated by multiplying the unrounded entered amount and the conversion rate and then rounding to the minimum accountable unit and currency

precision. Financial Services Accounting Hub stores unrounded and rounded entered amounts, as well as unrounded and rounded accounted amounts to provide an audit trail.

If the entered currency is the same as the ledger currency, the accounted amount is the same as the entered amount. If the entered currency is different from the ledger currency, conversion information must be provided in the transaction object in order for Financial Services Accounting Hub to calculate the accounted amount.

The following rules apply to conversion information:

- Conversion rate types must be defined in General Ledger.

  See: Defining Conversion Rate Types, *Oracle General Ledger User Guide*

- If the conversion rate type is User, a conversion rate must be provided.

- If the conversion rate type is not User, a conversion date must be provided.

  Financial Services Accounting Hub uses this rate type and conversion date to find the conversion rate defined in the General Ledger Daily Rates window.

  See: Entering Daily Rates, *Oracle General Ledger User Guide*

Override Calculated Accounted Amount

If Financial Services Accounting Hub is configured to calculate accounted amounts for a specific event class and there is a need to override the calculated accounted amount, users can map the Override Calculated Accounted Amount accounting attribute to a source and provide the accounted amount in the transaction object. If the value for this accounting attribute is Y for yes and the accounted amount is provided in the transaction object, Financial Services Accounting Hub uses the provided accounted amount. If the value for this accounting attribute is not Y and the accounted amount is provided in the transaction object, the provided accounted amount is ignored.

Users can only map the Override Calculated Accounted accounting attribute to a source if the Calculate Accounted Amounts option is enabled for the event class.

Reporting Currencies

Financial Services Accounting Hub calculates accounted amounts for subledger level reporting currencies if the Calculate Reporting Currency Amounts option is selected when registering the subledger application in Financial Services Accounting Hub. If the Calculate Reporting Currency Amounts option is not selected for the subledger application and the subledger level reporting currency is defined, implementers must provide the reporting currencies amount and conversion information for the transactions in the transaction objects. For example, if there is only one subledger level reporting currency associated with the primary ledger and the Calculate Reporting Currency Amounts option is not enabled for the application, then there should be two rows in the transaction object for each distribution with a different ledger identifier: one

for the primary ledger and one for the reporting currency.

See: Registering Subledger Applications, page 4-7

Calculation of Gain or Loss Amounts

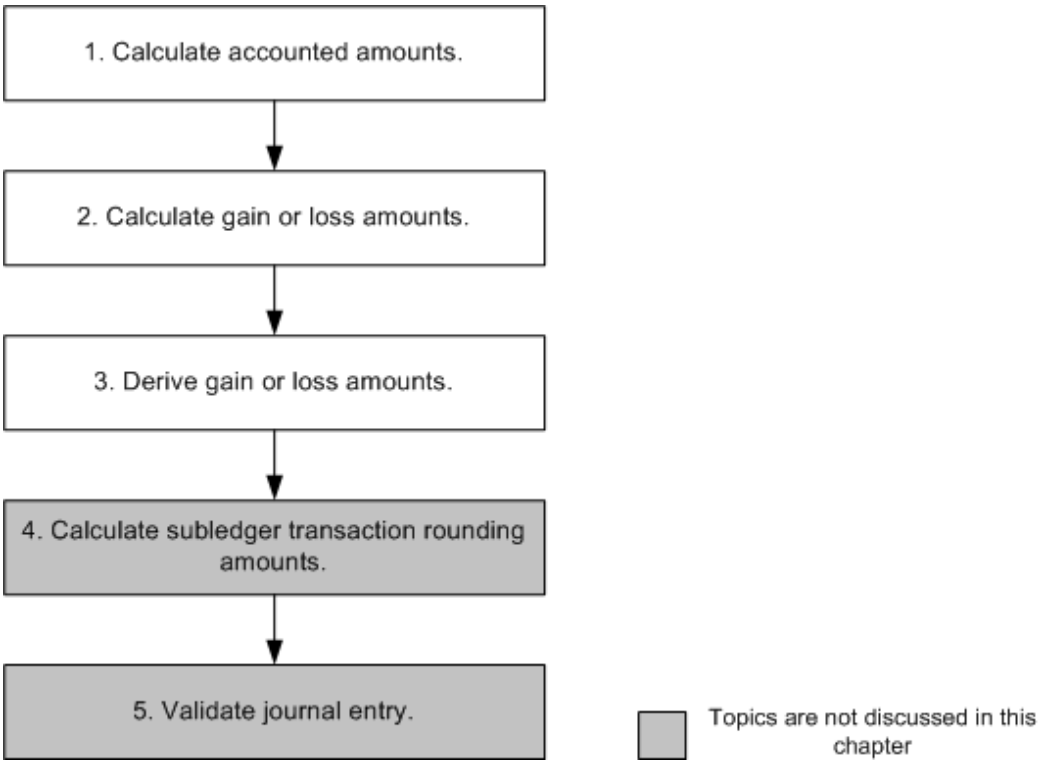
Gain or loss occurs when a transaction is entered in a currency other than the ledger currency and the conversion rate changes between the time different accounting events are triggered. The Financial Services Accounting Hub calculates the gain or loss amounts if the Calculate Accounted Amounts option is enabled for the event class and if a journal line type with Gain or Loss side is assigned to the journal line definition used to create accounting. When Calculate Accounted Amount is enabled, Calculate Gain or Loss Amounts must also be enabled. However, Calculate Gain or Loss Amounts can also be enabled if Calculate Accounted Amount is disabled. Depending on which options are selected, certain accounting attributes must have a value.

See: Step 9, To Set Up Accounting Event Class Options, page 5-4

The Financial Services Accounting Hub calculates gain or loss amounts by subtracting the unrounded accounted credit amount from the unrounded accounted debit amount for actual balance types. If the result is a positive amount, a gain line is created; otherwise, a loss line is created. The gain or loss journal line has a zero entered amount on the same side, debit or credit, as the accounted amount. Reversal of a gain or loss line does not result in revaluation of gain or loss. The derivation of Accounting Flexfields for gain or loss journal entry lines does not impact lines generated as the result of an accounting reversal. Lines generated as the result of an accounting reversal must always use the Accounting Flexfield and conversion rate of the reversed lines.

The Accounting Program Process Flow for Gain or Loss Amount Calculation

The diagram below shows the Accounting Program process flow for gain or loss amount calculation and is described in the succeeding text.



The Accounting Program process flow for gain or loss calculation is as follows:

1. Calculate accounted amounts.

   For all temporary journal lines for debit and credit journal line types, the Accounting Program calculates unrounded accounted amounts using the conversion information provided if Calculate Accounted Amounts is selected.

2. Calculate gain or loss amounts.

   For all temporary journal lines for gain or loss journal line types, the Accounting Program calculates gain or loss as the difference between the unrounded accounted amounts if Calculate Gain or Loss Amounts is enabled.

3. Derive gain or loss accounts.

   The Accounting Program determines whether it is a gain or loss and derives the Accounting Flexfield based on whether the journal line type is defined. If the gain/loss journal line type is defined, the value assigned to the journal line type is used to calculate the gain/loss. If the gain/loss journal line type is not defined, the gain/loss account assigned to the Exchange Gain Account and Exchange Loss Account accounting attributes is used.

4. Calculate subledger transaction rounding amounts.

   See: Transaction Rounding Reference Accounting Attribute, page 3-38

5. Validate journal entry.

See:

- Subledger Journal Entry Line Validation, *Oracle Subledger Accounting Implementation Guide*
- Subledger Journal Entry Balancing Rules, *Oracle Subledger Accounting Implementation Guide*
- Gain/Loss Accounting Attributes, page 3-25

16

Transaction Account Builder

Transaction Account Builder Overview

Use the Transaction Account Builder (TAB) to derive default accounts for their transactions using sources defined in the Accounting Methods Builder (AMB).

Derive accounting codes from the TAB and the AMB. Use the TAB to derive default accounts for transactions before they are accounted. Use the AMB to generate the accounts that appear in the accounting.

> **Note:** TAB only derives default accounts for transactions. These accounts may not be the ones that appear on the subledger journal entries since these are generated by the Create Accounting program based on the application accounting definitions.

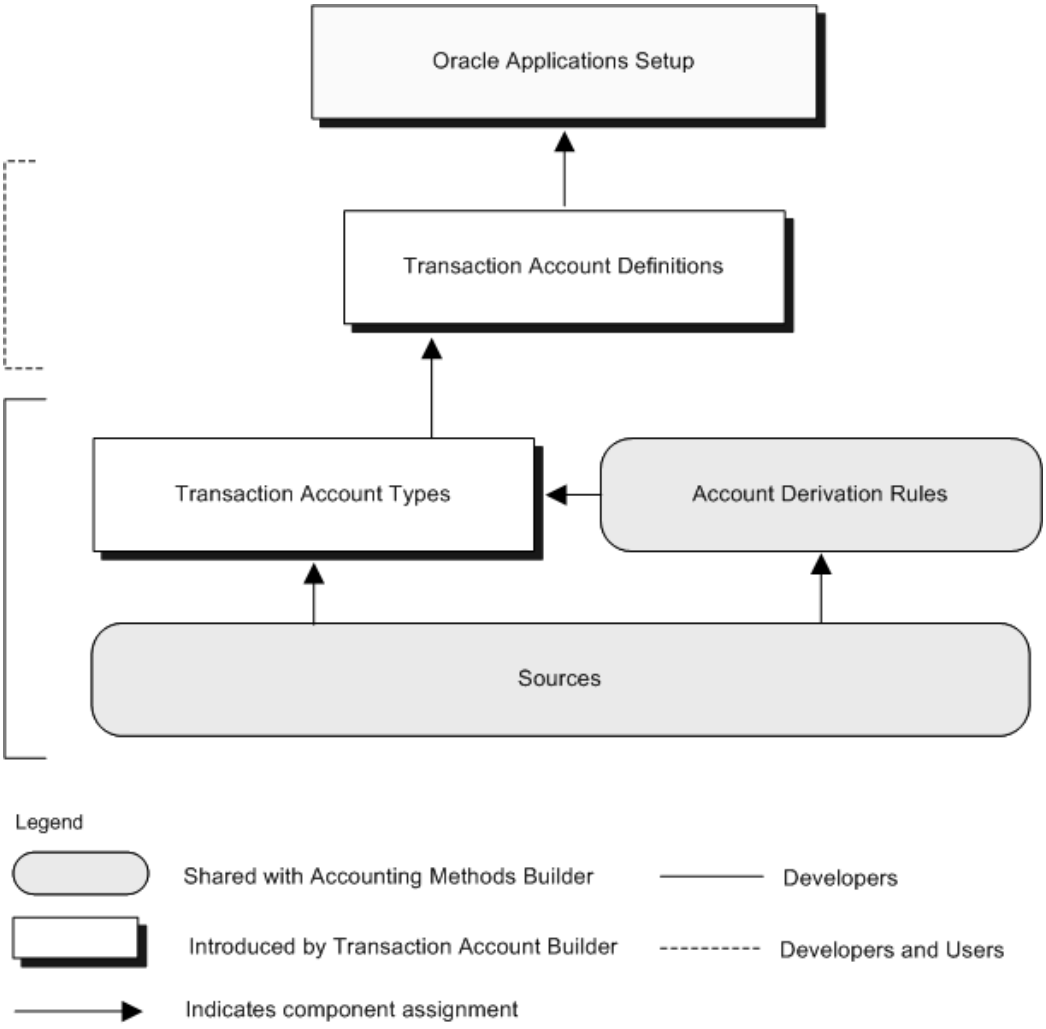
If the application does not allow modification of the default accounts for a transaction before it is accounted, you must use the AMB.

See: Accounting Methods Builder Introduction, *Oracle Subledger Accounting Implementation Guide*

Transaction Account Builder Components

The Transaction Account Builder Components figure below shows the components that comprise the TAB and is described in the subsequent text. Sources and account derivation rules are shared with the AMB, while transaction account types and transaction account definitions are specific to the TAB.

Transaction Account Builder Components



Implementers can assign sources to transaction account types. Implementers and users can use these sources to derive accounts for a transaction account type. For example, while the bank asset account can be used to derive the cash account for a payment, it might not be useful when deriving the invoice price variance account for a matched invoice. Consequently, the source for the bank asset account is not assigned to the transaction account type for the invoice price variance.

Examples of possible transaction account types for Receivables transactions are Revenue, Tax, Receivable, Freight, Unearned Revenue, Unbilled Receivable, and Bills Receivable. Examples of transaction account types for receipts are Cash, Unidentified Receipts, Unapplied Receipts, On Account Receipts, Earned Discounts, Unearned Discounts, Remitted Receipts, and Factored Receipts.

See: Step 2: Define Transaction Account Types, page 16-4

Implementers and users can assign account derivation rules to transaction account

types to create a transaction account definition. These definitions can be assigned as part of the applications setup.

See: Step 2: Create Transaction Account Definitions (Optional), *Oracle Subledger Accounting Implementation Guide*

Account Validation and Dynamic Insertion

When calling the TAB, you can pass the transaction account definition, chart of accounts, transaction account type, and source values from the transaction. Based on these values, the TAB builds an Accounting Flexfield combination and checks whether it is valid for the chart of accounts. If the combination does not exist and dynamic insertion is enabled for the chart of accounts, a new code combination is created. The TAB returns the code combination identifier for the Accounting Flexfield combination or an error code when appropriate.

The TAB does not make any security validation. This is part of transaction security and should be handled separately by each application. Cross-validation rules are only examined for new combinations.

Validate Transaction Account Definitions Program

This program generates the code to derive transaction accounts based on the transaction accounting definition.

Transaction Account Builder Setup Process

Note: Transaction account types must be defined for your applications during Financial Services Accounting Hub implementation. Transaction Account Types for Oracle applications are seeded and you cannot create new or modify existing transaction account type.

The TAB setup steps are as follows:

- Step 1: Define Sources, page 16-4
- Step 2: Define Transaction Account Types, page 16-4
- Step 3: Generate TAB API Objects, page 16-6
- Step 4: Create Account Derivation Rules (Optional), page 16-6
- Step 5: Create Transaction Account Definitions (Optional), page 16-6
- Step 6: Modify Application Setup to Support TAB, page 16-6

- Step 7: Call the Transaction Account Builder API, page 16-6
- Step 8: Using the Transaction Accounts in the Accounting Methods Builder (Optional), page 16-7

See: Transaction Account Builder Setup Process, *Oracle Subledger Accounting Implementation Guide*

Step 1: Define Sources

The TAB shares the sources that are defined in the AMB. You can select the sources that are available to derive accounts for each transaction type. Ensure that all potential sources of account derivation rules for transactions are defined in the AMB.

Additionally, define sources for all the transaction accounts defaulted by TAB, so that users can use these accounts for their account derivation rules in the AMB.

See: Introduction to Events and Sources, page 1-3

Step 2: Define Transaction Account Types

The TAB enables you to define the transaction account types for your applications. The transaction account types component enables you to categorize the different accounts that are generated for your transactions. Accounts that require a consistent derivation throughout the application should share the same transaction account type.

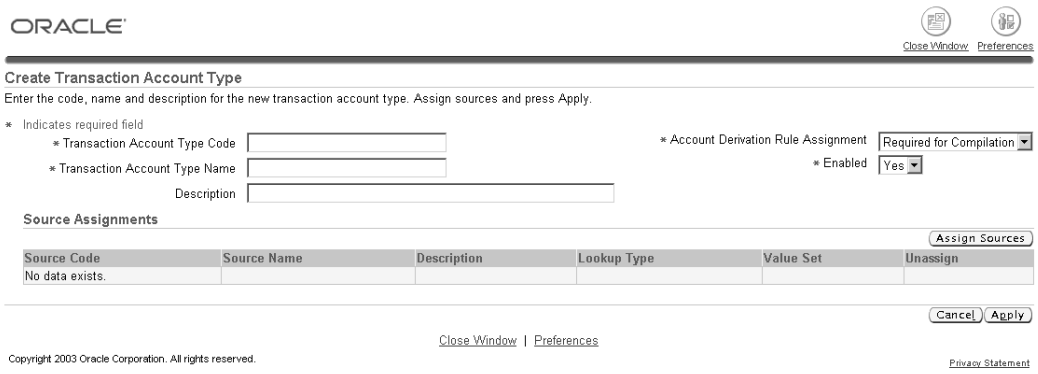
You can assign sources to transaction account types. Sources used by the TAB need not be Accounting Flexfields. When assigning sources to transaction account types, implementers should ensure that they include all sources that the users could be interested in using for deriving accounts for their transactions. This allows them to select the sources that can be used to derive accounts for each transaction account type. For example, in Receivables, assign the sources Salesperson Revenue Account, Transaction Type Revenue Account, and System Options Revenue Account to the Revenue transaction account type.

Decide whether a transaction account type requires an account derivation rule assignment. For example, in Receivables, the account derivation rule assignment for the unearned revenue account can be optional as this account is only required for invoices with rules.

See: Transaction Account Builder Components, page 16-1

Creating Transaction Account Types

In the Create Transaction Account Types page, define and maintain the types of accounts that are derived for your transactions and select the sources that are available to derive accounts for each transaction account type.



Selected Fields and Buttons in the Create Transaction Account Type Page and the Assign Sources Page

Field or Button	Description
Account Derivation Rule Assignment	Indicates whether account derivation rules must be assigned to the transaction account type before the transaction account definition can be successfully validated Account derivation rule assignments should be required for transaction account types that are likely to be used by most users. For example, in Receivables, the receivable and revenue accounts are required for every transaction; therefore, account derivation rules assignments for the corresponding transaction account types should be required for validation. On the other hand, the unearned revenue and unbilled receivable accounts are only required if the user is using invoices with rules; therefore, account derivation rules for the corresponding transaction account types could be optional for validation.
Enabled	Select to make available for assignment in transaction account definitions.
Assign Sources	Opens the Assign Sources page. Execute a search for sources.
Assign to Transaction Account Type	Assigns selected source to transaction account type

Step 3: Generate TAB API Objects

After creating the transaction account types for your application, run the Transaction Account Builder - Generate API program that creates the database objects required by the TAB API for your application.

Step 4: Create Account Derivation Rules (Optional)

If seeding transaction account definitions, you are required to create account derivation rules for these seeded definitions. When defining an account derivation rule, ensure that all sources used by the account derivation rule are assigned to the transaction account type to which it is assigned.

See: Account Derivation Rules, *Oracle Subledger Accounting Implementation Guide*

Step 5: Create Transaction Account Definitions (Optional)

You can seed transaction account definitions for your application. This expedites the setup process as users can assign these definitions as part of the application setup.

See: Step 2: Create Transaction Account Definitions, *Oracle Subledger Accounting Implementation Guide*

Step 6: Modify Application Setup to Support TAB

You are required to include a setup step enabling users to assign transaction account definitions.

Depending on the application, you can use a profile option or modify an existing window, such as Payables Options in Payables or System Options in Receivables, to make this assignment.

Step 7: Call the Transaction Account Builder API

Use the Transaction Account Builder API to call the TAB from your application. You are required to modify your application so that the Transaction Account Builder API is called every time a default account is required for a transaction. The following information needs to be passed to this API:

- Application
- Chart of accounts
- Transaction account definition
- Transaction account type

- Values for all sources assigned to the transaction account type

The TAB uses this information and the account derivation rules associated with the transaction account type to produce an Accounting Flexfield combination for the chart of accounts. If the combination does not exist and dynamic insertion is enabled for the chart of accounts, the TAB creates the combination.

Step 8: Using the Transaction Accounts in the Accounting Methods Builder (Optional)

You can use the TAB accounts in your subledger journal entries since they can be defined as a standard source and made available in the transaction objects.

See: Step 5: Using the Transaction Account Types as Default Accounts for the AMB, *Oracle Subledger Accounting Implementation Guide*

17

Application Accounting Definitions Loader

Application Accounting Definitions (AAD) Loader Overview

The Applications Accounting Definitions (AAD) Loader enables users to import and export application accounting definitions and journal entry setups between the file system and database instances. The AAD Loader also supports concurrent development and version control of the application accounting definitions.

In this chapter, seeded application accounting definitions refer to the ones created by Oracle implementers and shipped to customers. User-defined application accounting definitions refer to the ones created by implementers at the customer installation.

Version Control Overview

The AAD Loader satisfies the following version control requirements:

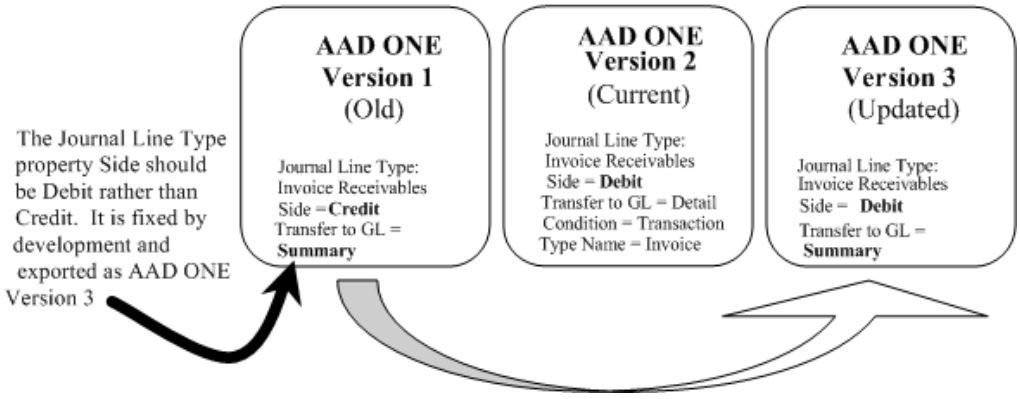
- Ensures version consistency for all the journal entry setups assigned to an application accounting definition
- Maintains consistency between user application accounting definitions and Oracle application accounting definitions for all shared journal entry steps
- Restores previous versions of application accounting definitions

All application accounting definitions, mapping sets, account derivation rules, and supporting references are stamped with a version number. The version history is recorded in the database. The AAD Loader records whether a version change is a result of leapfrogging. When an application accounting definition is updated based on a version earlier than its latest version, the result is defined as a leapfrog version. Leapfrogging is commonly used when implementers fix bugs on earlier versions. The version control mechanism ensures consistency of the application accounting definitions and journal entry setups during the import and export processes.

In the example described in the leapfrogging figure and table below, the latest version

of AAD ONE in development is version 2. A customer using AAD ONE Version 1 realizes that the property Side should be Debit rather than Credit for Journal Line Type Invoice Receivables. However, the customer does not want all the changes in version 2, namely the Transfer to GL property equals Detail and the added Condition property. The fix is made to version 1 and delivered to the customer as version 3. An updated version of AAD ONE Version 1 leapfrogs over the current AAD ONE Version 2 to become AAD ONE Version 3. The current version in development is now AAD ONE Version 3.

Leapfrogging Example



The table below describes the leapfrogging example.

AAD One Version 1 (Old)	AAD One Version 2 (Current)	AAD One Version 3 (Updated)
Journal Line Type: Invoice Receivables	Journal Line Type: Invoice Receivables	Journal Line Type: Invoice Receivables
Side = Credit	Side = Debit	Side = Debit
Transfer to GL = Summary	Transfer to GL = Detail	Transfer to GL = Summary
	Condition = Transaction	
	Type Name = Invoice	

See:

- Version Control: Application Accounting Definitions, page 17-6
- Version Control: Mapping Sets, Account Derivation Rules, and Supporting References, page 17-6

Installation, Patching, and Migration

This section includes the following parts:

Application Accounting Definitions Data Delivery

All application accounting definitions and journal entry setups for an application are delivered or imported using a single data file. The file includes only application accounting definitions and journal entry setups of an application. Other seed data, such as sources and event entities, classes, and types are delivered through regular seed data mechanisms.

Import Application Accounting Definitions

Application accounting definitions are imported using the Import Application Accounting Definitions concurrent program. This program imports the application accounting definitions from a data file to the Accounting Methods Builder (AMB) context specified in the SLA: Accounting Methods Builder Context profile option and produces a report of the results.

See: SLA: Accounting Methods Builder Context, *Oracle Subledger Accounting Implementation Guide*

When running the Import Application Accounting Definitions concurrent program, users specify whether to run the merge analysis, merge, or overwrite processes.

Import Application Accounting Definitions Program Parameters Description

The table below describes the parameters for the Import Application Accounting Definitions program.

Import Application Accounting Definitions Program Parameters

Parameter	Description
Accounting Methods Builder Context	AMB context for the application accounting definitions to be imported; defaulted from the SLA: Accounting Methods Builder Context profile option; required
Source Data File	Full path name, including the .ldt file name, of the data file containing the application accounting definitions to be imported; required Path name example: /home/jdoe/out/APAAD.ldt **Note:** At least one of the three source path names must be entered.
Source File Pathname for Base Application	Valid source file path name
Source File Pathname for Budgetary Control	Valid source file path name
Merge Analysis Only	Determines whether merge analysis is performed to the imported application accounting definitions. Default is Yes.
Batch Name	User-entered merge analysis report name
Import Option	Enabled and required only if the Merge Analysis Only parameter is set to No; indicates whether the application accounting definitions from the data file should be merged or overwritten to the database. Default is Merge.
Validate	Enabled and required only if the Merge Analysis Only parameter is set to No; determines whether the Validate Application Accounting Definitions concurrent program should be submitted to validate all the imported application accounting definitions. Default is Yes.

Parameter	Description
Force Overwrite	Enabled and required only if the Merge Analysis Only parameter is set to No and the Import Option is Overwrite; indicates whether the Financial Services Accounting Hub should allow an application accounting definition to be overwritten from the data file to the database if the version in the data file is lower than the version in the database. Default is No.

Import Application Accounting Definitions Report

When the Import Application Accounting Definitions program is completed successfully, a report highlighting all the imported application accounting definitions and their versions as well as all the errors that occurred during the import process is produced.

Application Accounting Definitions Merge Analysis

Merge Validations

The decision to merge application accounting definitions and journal entry setups results in the copying of the specified application accounting definitions and journal entry setups from the file system to the database.

The table below describes the relationship between the type of user and the actions they can take with certain application accounting definitions. For example, an Oracle developer can merge and overwrite Oracle application accounting definitions while a customer can overwrite but cannot merge seeded application accounting definitions.

User Type	Seeded Application Accounting Definition	User-defined Application Accounting Definition
Oracle Developer	Merge Overwrite	Not Available
Customer	Overwrite	Merge, Overwrite

When merging seeded application accounting definitions from the file system to the database in a customer environment, a seeded application accounting definition is deleted from the database if it does not exist in the file system.

Merge Application Accounting Definitions

Users merge application accounting definitions by submitting the Import Application Accounting Definitions concurrent program with the Merge option.

> **Note:** When importing application accounting definitions using the Merge option, the AAD Loader merges the relationships between the subledger accounting methods and application accounting definitions that do not contain application accounting definition assignments for the application.

See: Merge Validations, page 17-5

Version Control: Application Accounting Definitions

When merging application accounting definitions from the file system to the database, the versions of the application accounting definitions are validated. Merge is allowed if the original version of the application accounting definition to be merged is higher than the current version in the database.

Version Control: Mappings Sets, Account Derivation Rules, and Supporting References

If the version of a mapping set, account derivation rule, or supporting reference in the data file is lower than that in the database, it is not merged into the database. The higher version is retained.

Merge Application Accounting Definitions Report

When completed successfully, the Import Application Accounting Definitions program produces a report highlighting all the merged application accounting definitions, mapping sets, account derivation rules, and supporting references setups as well as all the errors that occurred during the merge process.

Overwrite Application Accounting Definitions

Users overwrite application accounting definitions by submitting the Import Application Accounting Definitions program with the Overwrite option.

See: Import Application Accounting Definitions Program Description, page 17-3

Overwrite Validations

The overwrite program copies all the application accounting definitions from the file system to the database. The previous copies of application accounting definitions in the database are replaced.

> **Note:** The Force Overwrite parameter in the Import Application Accounting Definitions program must be set to Yes in order to

overwrite all application accounting definitions and journal entry setups regardless of their version number in the database.

Overwrite Application Accounting Definitions Report

When completed successfully, the Import Application Accounting Definitions program produces a report highlighting all the overwritten application accounting definitions, mapping sets, account derivation rules, and supporting references setups as well as all the errors that occurred during the overwrite process.

Export Application Accounting Definitions

The Export Application Accounting Definitions program exports all application accounting definitions of an application from a database to the file system and produces a report of the results. All application accounting definitions and journal entry setups for an application are exported to the same data file. When the application accounting definitions are exported, a new version is stamped on the application accounting definitions, mapping sets, account derivation rules, and supporting references referenced by exported application accounting definitions.

Export Application Accounting Definitions Program Parameters

The table below describes the parameters for the Export Application Accounting Definitions program.

Export Application Accounting Definitions Program Parameters

Parameter	Description
Account Method Builder Context	AMB context for the application accounting definitions to be imported; defaulted from the SLA Accounting Methods Builder Context profile option; required

Parameter	Description
Destination File Path	Full path name, including the .ldt file name, of the file system where the application accounting definitions are to be exported; required Path name example: /home/jdoe/out/APAAD.ldt **Note:** At least one of the three destination file path names must be entered.
Destination File Pathname for Base Application	Valid file path name for the base application
Destination File Pathname for Budgetary Control	Valid path name for the federal budgetary control
Versioning Mode	Indicates whether the exported application accounting definitions are a result of leapfrogging. Select Standard if the AAD to be exported is based on the latest version; select Leapfrog if the AAD to be exported is not based on the latest version; or select Supersede if the AAD to be exported is not based on the latest version and is not a leapfrog. Default is Standard.
User Version	User-assigned version; optional
Export Comment	User-entered export comments; optional

Export Validations: Application Accounting Definitions

All application accounting definitions that belong to the same owner, Oracle or User, and share any journal entry setups are considered part of the same application accounting definition group. All application accounting definitions in the same group are exported with the same version number.

If any application accounting definition in a group is modified, all the application accounting definitions within the group are exported with a new version number. The new version number is the application accounting definition in the group with the highest version number plus one.

Export Validations: Mapping Sets, Account Derivation Rules, and Supporting References

The following validations apply to mappings sets, account derivation rules, and supporting references upon export:

- If mapping sets, account derivation rules, or supporting references are modified, they are exported with the highest version of the mapping set or account derivation rule in the history table plus one.

- If mapping sets, account derivation rules, or supporting references are not modified, they are exported with the same version number.

  If any of the unmodified mapping sets or supporting references result from a forced overwrite, they are exported with a new version number based on the highest version of the mapping set in the history table plus one.

Export Application Accounting Definitions Report

When completed successfully, the Export Application Accounting Definitions program produces a report highlighting all the exported application accounting definitions as well as all the errors that occurred during the export process.

Application Accounting Definitions Migration

The AAD Loader does not support the direct migration of application accounting definitions from one instance to another. Instead, the migration of application accounting definitions is supported by the following operations:

- Application accounting definitions are exported from the source instance into a data file.

- Application accounting definitions are imported from the data file to the destination instance with the option to overwrite all application accounting definitions from the staging area to the working area of an AMB context.

Applications Accounting Definitions Concurrent Development

Users can create multiple AMB contexts to support multiple implementers working concurrently.

This section includes the following parts:

- Development by a Single Implementer, page 17-10

- Concurrent Development, page 17-10

- Development by a Single Implementer at a Customer Site, page 17-11

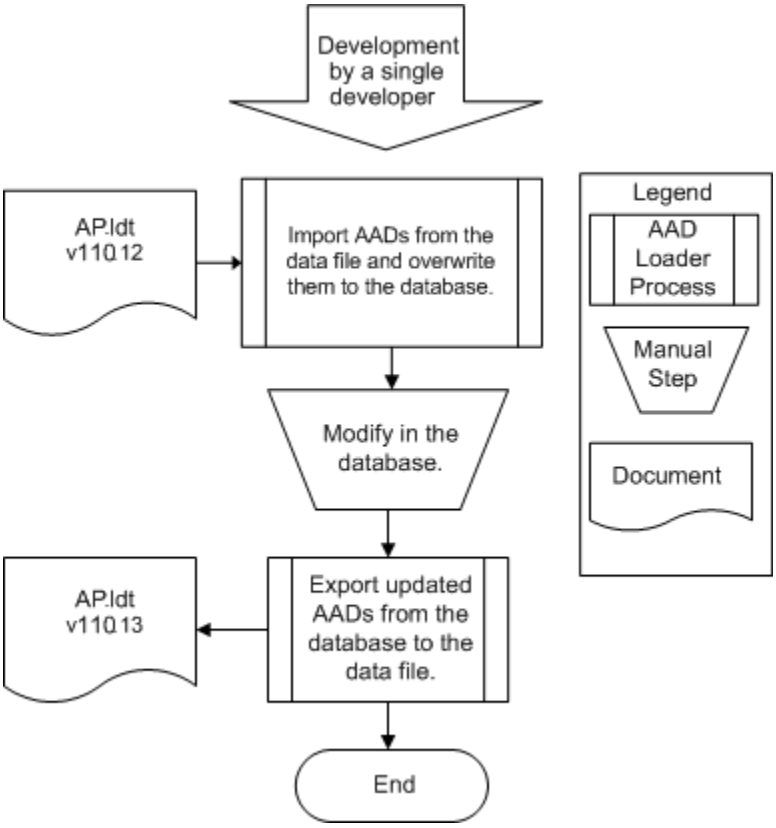
- Concurrent Development by Multiple Implementers at a Customer Site, page 17-12

Development by a Single Implementer

Implementers use the AAD Loader to import and export application accounting definitions between the data file and the database instances.

In the example shown in the figure below, a single Implementer imports a data file into the AMB context and completely overwrites the application accounting definitions in the database with the application accounting definitions from the imported data file. The application accounting definitions are then exported into a new data file.

Development by a Single Implementer Example



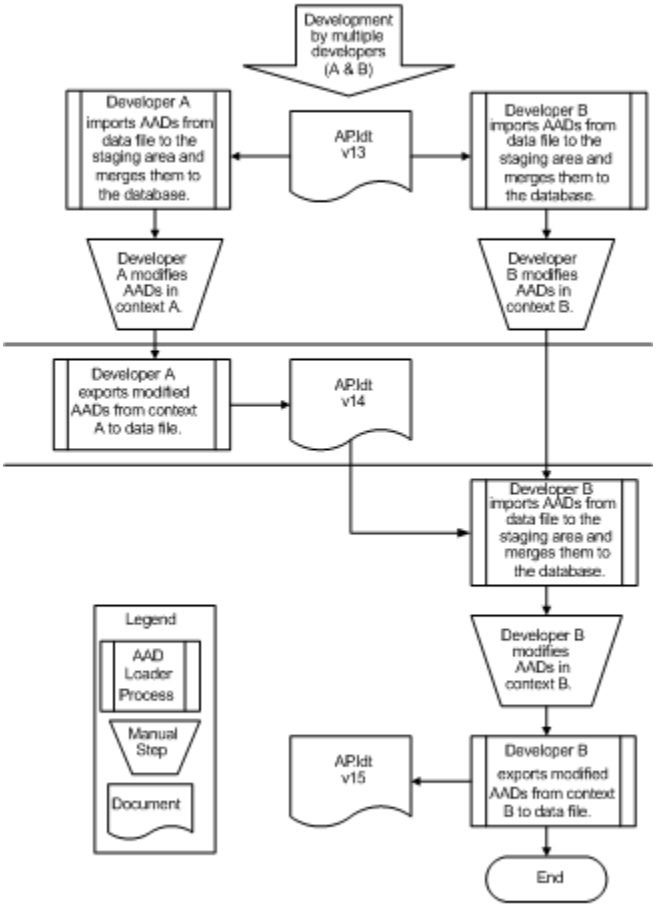
Concurrent Development

Concurrent development is possible because the AMB enables implementers to simultaneously work on the same application accounting definitions in different AMB contexts. Data files with updated versions of application accounting definitions are then imported and merged into specified AMB contexts to load the updated application accounting definitions into the database.

In the example shown in the figure below, multiple implementers import a data file into

their respective AMB contexts and merge the application accounting definitions in their context with the application accounting definitions from the imported data file. Implementer A completes the work and exports the updated application accounting definitions into a new data file. Implementer B imports the data file and merges the updated application accounting definitions into context B. The merged application accounting definitions are then exported along with changes made by Implementer B into a new data file.

Concurrent Development Example

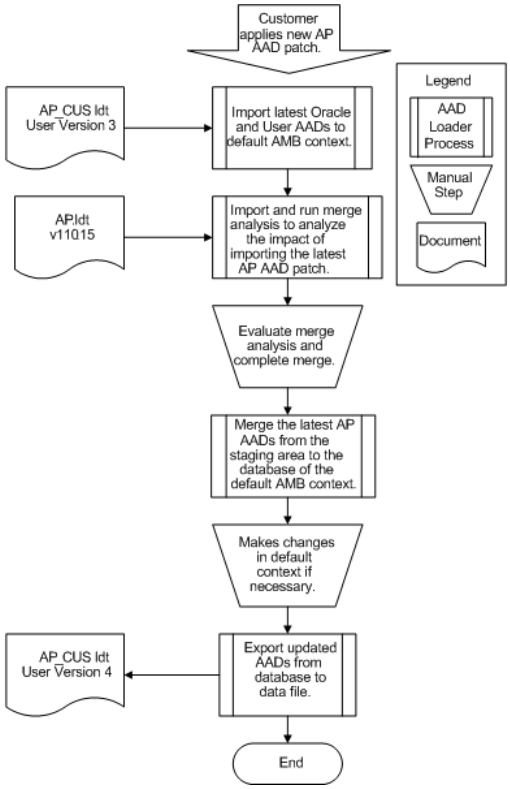


Development by a Single Implementer at a Customer Site

The AAD Loader facilitates development of customer-specific application accounting definitions at the customer site by providing the same import, export, and merge utilities available to Oracle implementers.

In the example shown in the figure below, a patch with seeded application accounting definitions is applied at the customer site. The latest versions of all the application accounting definitions currently in use by the customer are imported into the working area of a default AMB context before applying the patch. The data file from the patch is imported and a merge analysis is run highlighting the impact of the new seeded application accounting definitions before merging the application accounting definitions. The merged application accounting definitions are then exported along with changes made by the implementer into a new data file.

Development by a Single Implementer at a Customer Site Example

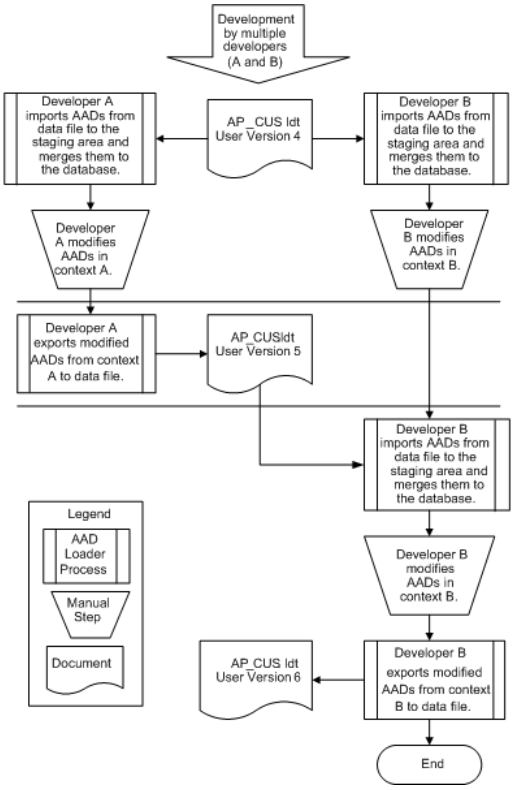


Concurrent Development at a Customer Site

The AAD Loader offers the same concurrent development at the customer site.

In the example shown in the figure below, multiple implementers at the customer site import a data file into their respective AMB contexts and merge the application accounting definitions in their working areas with the application accounting definitions from the imported data file. Implementer A completes the work and exports the updated application accounting definitions into a new data file. Implementer B imports the data file and merges the updated application accounting definitions into the working area in context B. The merge application accounting definitions are then exported along with changes made by Implementer B into a new data file.

Concurrent Development at a Customer Site Example



Index

A

B

C

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

W