Oracle® Healthcare Transaction Base

Implementation Guide

Release 11*i*

Part No. B13734-01

August 2004

ORACLE®

Oracle Healthcare Transaction Base Implementation Guide, Release 11*i*

Part No. B13734-01

Primary Author: Mike Cowan

Contributing Authors: Marita Isidore, Manu Kumar

Contributors: Shengi Cheng, John Hatem, Sandy Hoang, Ravichandra Hothur, Anand Inumpudi, Flora Kidani, Valerie Kirk, Ben Lee, Patrick Loyd, Gloria Nunez, Tom Oniki, Balan Ramasamy, Shelly Qian, Cindy Satero, Andrea Sim, Pauline Troiano

Contents

List of Figures

List of Tables

Send Us Your Comments

Oracle Healthcare Transaction Base Implementation Guide, Release 11*i*

Part No. B13734-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appsdoc_us@oracle.com
- FAX: (650) 506-7200 Attn: Oracle Applications Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle Applications Documentation Manager
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Welcome to Release 11*i* of the Oracle Healthcare Transaction Base Implementation Guide.

This guide is intended for users and systems professionals involved with the implementation and configuration of Oracle Healthcare Transaction Base, including the following:

- Implementation Consultants
- System Administrators
- Security Administrators
- Healthcare Application Developers

> **Note:** This guide describes the implementation and configuration procedures for Oracle Healthcare Transaction Base; refer to *Oracle Javadoc for HTB* to understand its full functionality.

This guide assumes that you have the following general skill sets:

- Knowledge of the Java Language (J2EE platform), Oracle PL/SQL, and Oracle Relational Database 9*i.*
- Implementation experience with Oracle E-Business Suite Release 11*i* and Oracle Workflow.
- Administration experience with Oracle9*i*AS, and knowledge of Oracle Containers for J2EE (OC4J), and EJB deployment.

- Familiarity with Oracle Healthcare Transaction Base.

See: Other Information Sources for more information about related Oracle Applications.

How To Use This Guide

This document contains information required to implement and configure Oracle Healthcare Transaction Base*:*

Part I, Getting Started

Chapter 1, Introduction

Introduces Oracle Healthcare Transaction Base and describes its features.

Chapter 2, Before You Begin

Summarizes which applications and Oracle Technology Stack components must be installed, implemented, and verified before implementing Oracle Healthcare Transaction Base.

Part II, Implementing Oracle Healthcare Transaction Base

Chapter 3, Implementation Overview

Provides an overview of the HTB implementation process, including the implementation task sequence, with cross-references to specific implementation procedures described in Chapter 4.

Chapter 4, Implementation Tasks: HTB Platform

Describes each implementation procedure for the HTB Platform, organized in implementation sequence. These procedures are mapped directly by the implementation task description in Chapter 3. Implementation procedures for Payer Financial Management are described by Chapter 5.

Chapter 5, Implementation Tasks: Payer Financial Management

Provides an overview of Payer Financial Management (PFM), a component of Oracle Healthcare Transaction Base, including implementation procedures and PFM report descriptions. Also provides high-level guidance for the design and development of a customized extraction, transformation, and load process, which is required to transfer data from source transactional systems into the PFM financial repository.

Appendix J, Party Merge Procedures

Describes the Party Merge processes supported by HTB, including the process flow, details related to merged or transferred data, and any HTB validations that occur before completion of a process.

Appendix K, Act Definition Messaging Considerations

Describes act definition messaging considerations, including proposed act definitions that can be used in each message domain.

Appendix L, Clinical Business Services and ETS Concept Lists

Describes the use of ETS concept lists in Clinical Business Services.

Appendix M, Running Concurrent Programs

Describes how to run Oracle Concurrent programs using the Oracle Concurrent Manager. The procedure is the same for all Oracle Applications.

Appendix N, Abbreviations & Acronyms

Defines abbreviations and acronyms used throughout Oracle Healthcare applications.

Glossary

Contains the master glossary for Oracle Healthcare Applications.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. 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 additional information, visit the Oracle Accessibility Program Web site at:

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

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, 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, JAWS 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.

Other Information Sources

You can choose from many sources of information, including documentation, training, and support services, to increase your knowledge and understanding of Oracle Healthcare Transaction Base.

If this guide refers you to other Oracle Applications documentation, use only the Release 11*i* versions of those documents.

Online Documentation

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

- **PDF Documentation-** See the Documentation CD provided with each release for current PDF documentation for your product. This Documentation CD is also available on Oracle*MetaLink* and is updated frequently.
- **11i Release Content Document -** Refer to the Release Content Document for new features listed for each release. The Release Content Document is available on Oracle*MetaLink*.
- **About document -** Refer to the About document for patches that you have installed to learn about new documentation or documentation patches that you can download. The new About document is available on Oracle*MetaLink*.

Related Documentation

Oracle Healthcare Transaction Base shares business and setup information with other Oracle Applications products. You may want to refer to other product documentation when you set up and use Oracle Healthcare Transaction Base.

You can view the guides online by reading from the Oracle Applications Document Library CD included in your media pack, or by using a Web browser with a URL provided by your System Administrator.

Documents Related to All Products

Oracle Applications User's Guide

This guide explains how to enter data, query, run reports, and navigate using the **Graphic User Interface** (GUI). This guide also includes information about setting user profiles, as well as running and reviewing reports and concurrent processes.

Installation and System Administration

Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before an installation of Oracle Applications. This guide also introduces the concepts behind Applications-wide features such as Business Intelligence (BIS), languages and character sets, and Self-Service Web Applications.

Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle Rapid Install, which minimizes the time to install Oracle Applications and the Oracle technology stack by automating many of the required steps. This guide contains instructions for using Oracle Rapid Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user guides and implementation guides.

Oracle Applications Implementation Wizard User Guide

If you are implementing more than one Oracle product, you can use the Oracle Applications Implementation Wizard to coordinate your setup activities. This guide describes how to use the wizard.

Upgrading Oracle Applications

Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11*i*. This guide describes the upgrade process and lists database and product-specific upgrade tasks. You must be either at Release 10.7

(NCA, SmartClient, or character mode) or Release 11.0 to upgrade to Release 11*i*. You cannot upgrade to Release 11*i* directly from releases prior to 10.7.

About Oracle Healthcare Family Pack D

This supplement to the Oracle Healthcare Transaction Base Implementation Guide describes known HTB issues and limitations at the time of release. Intended for HTB installers and developers building HTB-based applications, this document describes Oracle Healthcare Family Pack D, and provides instructions for applying patches, new and changed setup steps, and descriptions of software updates. You should review this document before commencing implementation of Oracle Healthcare Transaction Base. Refer to Oracle*eMetaLink* to obtain the most current version.

Maintaining Oracle Applications

Use this guide to help you run the various AD utilities, such as AutoUpgrade, AutoPatch, AD Administration, AD Controller, AD Relink, License Manager, and others. It contains how-to steps, screenshots, and other information that you need to run the AD utilities. This guide also provides information about maintaining the Oracle applications file system and database.

Oracle Applications System Administrator's Guide

This guide provides planning and reference information for the Oracle Applications System Administrator. It contains information about how to define security, customize menus and online help, and manage concurrent processing.

Oracle Alert User's Guide

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

Oracle Applications Developer's Guide

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

Oracle Applications User Interface Standards for Forms-Based Products

This guide contains the user interface (UI) standards followed by the Oracle Applications development staff. It describes the UI for Oracle Applications products and how to apply this UI to the design of an application based on Oracle Forms.

Other Implementation Documentation

Oracle Applications Product Update Notes

Use this guide as a reference for upgrading an installation of Oracle Applications. It provides a history of the changes to individual Oracle Applications products between Release 11.0 and Release 11*i*. It includes new features, enhancements, and changes made to database objects, profile options, and seed data for this interval.

Oracle Workflow Administrator's Guide

This guide explains how to complete the setup steps necessary for any Oracle Applications product that includes **workflow**-enabled processes, as well as how to monitor the progress of runtime workflow processes.

Oracle Workflow Developer's Guide

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

Oracle Workflow User's Guide

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

Oracle Workflow API Reference

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

Oracle Applications Flexfields Guide

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

Oracle eTechnical Reference Manuals

Each eTechnical 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, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. *Oracle eTRM for HTB* includes logical and physical data models for both ETS and HTB. It is available on Oracle*Metalink.*

Oracle Applications Message Manual

This HTML manual describes Oracle Applications messages.

Other Documentation Related to Oracle Healthcare Transaction Base

Oracle Javadoc for HTB

Derived from the HTB source code, this HTML document describes the public HTB Application Programming Interface (API), including all of the packages, interfaces, classes, and methods available to healthcare application developers using the HTB platform. This supplement to the Oracle Healthcare Transaction Base Implementation Guide is included on the HTB source code CD, and is also available on Oracl*eMetaLink*.

Oracle Healthcare Transaction Base Statement of Direction

The Statement of Direction describes the scope of the HTB Platform, including likely future enhancements, and provides a vision for the future direction and growth of the product. Intended for a general user audience and for those interested in the future direction of the product, it is available on Oracl*eMetaLink*.

HTB White Papers

Oracle White Papers describe topics of general interest to the HTB user community, and can be accessed on Oracl*eMetaLink*.

Training

Oracle offers training courses to help you and your staff master Oracle Applications and reach full productivity quickly. These courses are organized into functional learning paths, so you take only those courses appropriate to your job or area of responsibility. Courses are typically offered by Oracle University at one of our many education centers, or you may arrange for our trainers to teach at your facility. In addition, Oracle training professionals can tailor standard courses or develop

custom courses to meet your particular needs. Contact your Oracle Support Representative to determine which training solutions are available to support Oracle Healthcare Transaction Base.

Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Healthcare Transaction Base working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle9*i* server, and your hardware and software environment.

Oracle*MetaLink*

Oracl*eMetaLink* is your self-service support connection with web, telephone menu, and e-mail alternatives. Oracle supplies these technologies for your convenience, available 24 hours a day, 7 days a week. With Oracl*eMetaLin*k, you can obtain information and advice from technical libraries and forums, download patches, download the latest documentation, look at bug details, and create or update TARs. To use MetaLink, register at:

```
http://metalink.oracle.com
```

Alerts: You should check Oracl*eMetaLink* alerts before you begin to install or upgrade any of your Oracle Applications. Use the following path to navigate to the Alerts window:

```
Technical Libraries/ERP Applications/Applications Installation
and Upgrade/Alerts
```

Self-Service Toolkit: You can also find information by navigating to the Self-Service Toolkit window, using the following path:

```
Technical Libraries/ERP Applications/Applications Installation
and Upgrade
```

Related Healthcare Industry Publications

This section describes publications within the Healthcare industry that describe topics conceptually related to Oracle Healthcare Transaction Base—principally those associated with **Health Level Seven** (HL7), a healthcare standard for electronic data exchange developed to establish national standards for electronic healthcare transactions.

Notes Regarding HL7 Publications:

- **Health Level Seven, Inc.** asserts and retains copyright and intellectual property rights in all works contributed by its members and non-members, relating to all versions of the **Health Level Seven** (HL7) standards and related materials—including all of the publications referenced in this section.
- The HL7 publications listed in this section are only available in electronic form. You can print what you need from the form, although PDF or word documents are typically provided as well.
- In accordance with the HL7 Intellectual Property agreement, you can download and use these materials for your own purposes if your organization is an HL7 member and you properly attribute the source to HL7.
- Non-members can also purchase publications directly from the HL7 Bookstore at:

  `www.hl7.org/library/bookstore`

- See Also: `www.hl7.org`

Unified Service Action Model Version 2.6

This is the complete version of the **Unified Service Action Model** (USAM), which describes the basic structures of the HL7 **Reference Information Model** (RIM).

Of interest to the HTB user community, including both users and application developers, this publication provides background information about the core information model upon which HTB and its APIs are based.

Message Development Framework, 1999

Describes Version 3 methodology, including the v3 meta model, and how the RIM, the datatypes and the vocabulary are created and integrated. It describes how to apply constraints to **Reference Information Model**s to derive **Domain Message Information Model**s, and how to relate DMIMs, **Refined Message Information Model**s, **Hierarchical Message Description**s, and **Message Type**s.

Of interest to the HTB user community, including both users and application developers, this publication provides background information about the core v3 message development methodology within HTB. It also explains how the **Unified Modeling Language** is used to derive serialized information structures, such as messages from an object model, and it describes HL7 use of UML (v3 metamodel).

HL7 Ballots

HL7 ballots describe the Reference Information Model, Message Types, and vocabulary. The ballots contain current version reference material, including the following:

- Introduction / Backbone (how to navigate the ballot)
- Help Guide (general background for the current version)
- Abstract Datatypes Specification
- RIM Vocabulary
- XML Implementable Technology Specification
- The actual ballot content

Of interest to the HTB user community, including users and application developers, HL7 ballots provide additional background information about the core information model upon which HTB clinical, administrative, and financial application APIs are based. Although there have been some significant changes to the original USAM model over time, HL7 ballots contribute to the conceptual framework for HTB development.

Reference Information Model Composite Zip Files (RIM 0116-0119)

Includes complete RIM versions, with vocabulary, Microsoft Word document forms, and Microsoft Excel files containing UML diagrams. The first HTB release is based on RIM 0116. Microsoft Word and Microsoft Excel are products of Microsoft Corporation.

Of interest to the HTB user community, including users and application developers, this document set provides background information about the core information model upon which HTB and its APIs are based.

Rosetree RIM Browser and Message Development Tool

The Rosetree RIM Browser lets you read the RIM 116 MDB file—used for HTB Release 1.0—which lets you browse the RIM as a UML model, including RIM datatypes and vocabulary.

Of interest to the HTB user community, including users and application developers, this analytical tool is an invaluable aid to understanding the RIM.

Other HL7 Version 3 Model and Message Development Tools

Used with **Rosetree**, these tools let you do the following:

- Create HL7 UML models with the HL7 Visio toolset (requires Microsoft Visio 2000, a product of Microsoft Corporation).
- Validate such models against the RIM, and load them into Rosetree.
- Create **Hierarchical Message Description**s.
- Create **Message Type** and related conformance profiles.
- Create XML XSDs that express message content and structure.
- Generate actual message instances.

Of interest to the HTB user community, including users and application developers, this analytical tool is an invaluable aid to message development.

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 Oracle Applications 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.

About Oracle

Oracle develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 160 software modules for financial management, supply chain management, manufacturing, project systems, human resources and customer relationship management.

Oracle products are available for mainframes, minicomputers, personal computers, network computers and personal digital assistants, allowing organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified computing and information resource.

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 145 countries around the world.

Part I

Getting Started

This part of the Oracle Healthcare Transaction Base Implementation Guide describes how to get started with the implementation process, and contains the following chapters:

- Chapter 1, Introduction
- Chapter 2, Before You Begin

1

Introduction

This chapter introduces Oracle Healthcare Transaction Base and describes its features.

This chapter contains the following topics:

- Section 1.1, Introduction
- Section 1.2, Overview

1.1 Introduction

Oracle Healthcare Transaction Base (HTB) is a comprehensive data repository and service infrastructure that provides Independent Software Vendors, System Integrators, and provider organizations with a state of the art software platform that lets them build robust and scalable healthcare applications.

HTB consists of software components that centralize and consolidate patient, provider and clinical objects and business rules across the enterprise. It overcomes operational challenges faced by healthcare organizations, letting them:

- Manage business transactions consistently in the patient care process, thereby increasing the efficiency, quality and competitive edge of the organization.
- Maximize clinical data reuse and portability through the use of standards, enabling seamless integration and consistent implementation between variant healthcare information systems.
- Enable protocol-based decision support, based on a complete view of patient information.

1.1.1 Consistent Business Transactions

HTB provides a single definition of each business object (encounter, patients, providers, medical acts...) across the healthcare organization. This lets users create and update key patient information in a consistent manner. For example, patient encounter information can be shared by various authorized personnel across a healthcare organization—enabling the creation and maintenance of a consolidated electronic health record.

1.1.2 Clinical Data Reuse and Portability

In many contemporary healthcare organizations, individual departments maintain their own independent information systems. These systems typically operate autonomously and do not synchronize patient data—making it difficult to develop a consistent, integrated view of the patient.

HTB provides a unified data model based on the HL7 v3 **Reference Information Model** (RIM), combined with sophisticated terminology mediation services incorporating standard terminologies that enable caregivers to efficiently manage and synchronize patient information. This approach lets caregivers avoid time-consuming data entry in multiple data sources while integrating patient information.

1.1.3 Comprehensive Community View

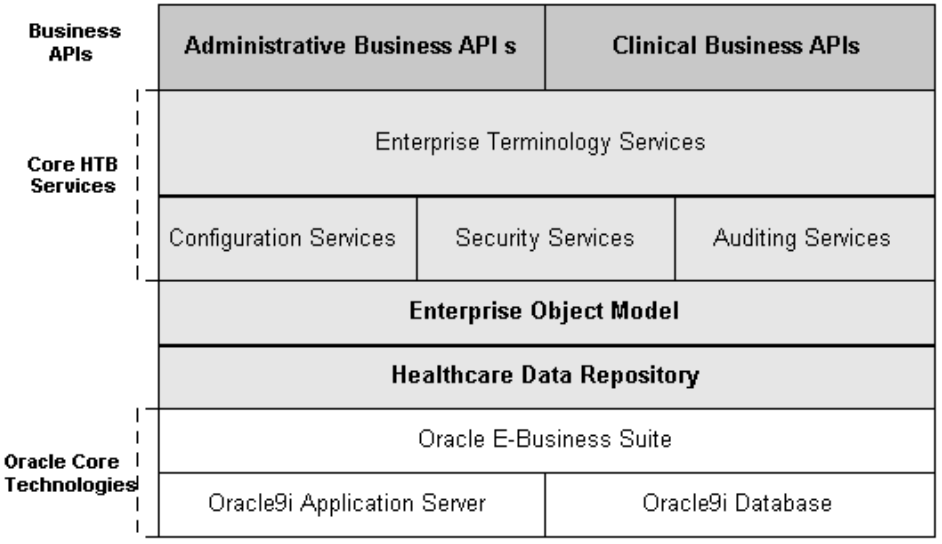
HTB lets multiple departments within a healthcare organization share data while maintaining a high level of autonomy. HTB provides multi-organization functionality that lets healthcare providers consistently manage and update central patient electronic healthcare records. As each department (or patient) touches a patient's specific data, the central patient record is updated accordingly. All organizations in a healthcare community can access relevant patient information (with appropriate patient consent), which translates into improved patient care, safety and reduced costs.

1.2 Overview

HTB consists of a set of services based on a foundation of selected Oracle core technologies. It includes software components that centralize and consolidate patient, provider, and clinical objects across the healthcare enterprise—providing unified access to a comprehensive healthcare infrastructure. The structure is a relational database implementation of the Reference Information Model that was developed for version 3 of the HL7 messaging standard and the corresponding vocabulary sets.

Figure 1–1 displays the HTB architecture:

Figure 1–1 HTB Architecture



Business logic in the functional domains (administrative, clinical) and core services are exposed through a Java-based Applications Programming Interface (API). The underlying HTB business logic can be extended as required for specific application functionality.

The HTB platform supports industry standards, such as HL7, HIPAA privacy regulations, and standard terminology sets such as SNOMED, CPT4, and ICD9. You

can use this platform to accelerate your migration to these industry standards while focusing on development of healthcare application functionality.

1.2.1 Business APIs

HTB Business APIs provide access to the content of the HTB data repository. They provide a thin layer of business logic, and depend upon the core HTB services.

For example, an API for ordering a drug issues a call to a security API to verify authorization, and issues a call to an Enterprise Terminology Service API to validate drug codes. The business domains supported in the initial HTB release include basic administrative functionality as well as management of orders and observations.

The core HTB services define a common service infrastructure for the development of functional components in healthcare applications. These services are exposed through a Java-based API. The core HTB service APIs provide the basis for the business functionality. They also support efficient development of secure and scalable applications on a normalized and secure date repository.

1.2.2 System Administration Applications

HTB platform services are managed through the following window-based administrative applications:

- Security Manager lets System Administrators define security policy and manage security services.
- Configuration Manager lets System Administrators manage HTB configuration services.
- ETS Workbench lets System Administrators manage enterprise terminology services.

1.2.3 Leveraging Existing Oracle Core Technologies

HTB uses Oracle core technologies, which provide high performance and scalability characteristics, and ensure seamless integration with other Oracle products that use the same technology foundation.

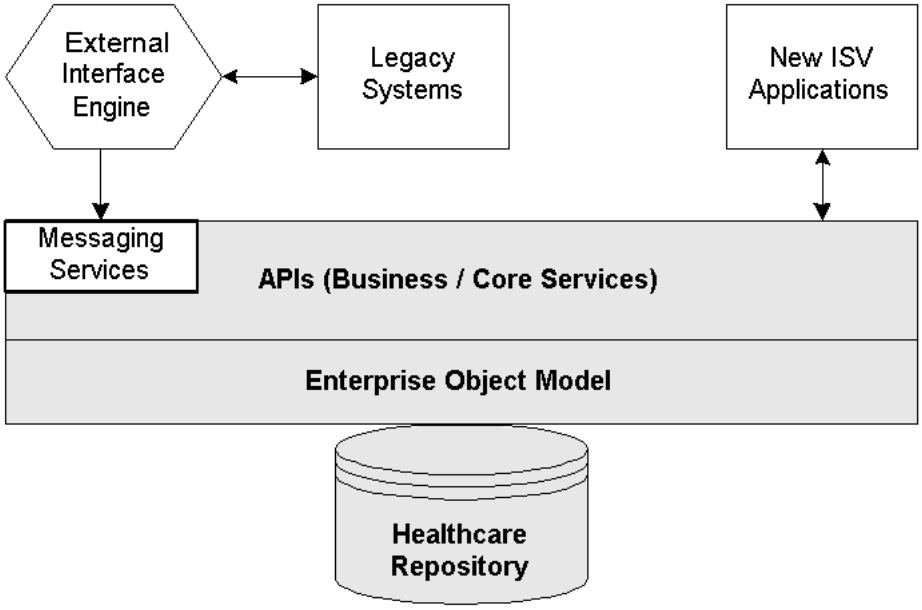
See Also: Section 2.1.1, Core Technology Foundation Components

1.2.4 HTB Usage Model

Figure 1–2 illustrates the usage model for the HTB platform:

- API-based integration for the development of new applications.
- Message-based integration with legacy systems.

Figure 1–2 HTB Usage Model



2

Before You Begin

This chapter summarizes the software components that must be installed, implemented, and verified before implementing the Oracle Healthcare Transaction Base.

This chapter contains the following topics:

2.1 Software Prerequisites

Before commencing the HTB implementation process, you must install and verify the software components described in the following sections, in the order listed:

- Section 2.1.1, Core Technology Foundation Components
- Section 2.1.2, Oracle E-Business Suite
- Section 2.1.3, Oracle Containers for J2EE
- Section 2.1.4, HTB EAR File Installation
- Section 2.1.5, Configuring your HTB Installation
- Section 2.1.6, Client Environment Library Requirements

2.1.1 Core Technology Foundation Components

Table 2–1 lists the core technology foundation components required to support implementation of Oracle Healthcare Transaction Base. To install and verify installation of these components, refer to the associated Oracle documentation for each component:

Table 2–1 HTB Technology Foundation Components

Tools	Components	Version
General Java Development	Oracle9*i* JDeveloper ▪ BC4J ▪ JDBC ▪ XML Developer's Kit for Java ▪ Enterprise JavaBean (EJB) ▪ Java Development Kit (JDK)	9i Production [9.0.3] ▪ 9i Production ▪ Java 2, v1.3, 9.0.1 ▪ 9i Production ▪ J2EE 1.3, EJB 2.0 ▪ v1.3.1
Oracle Enterprise Manager[1]	...	2.2
Oracle Workflow[1]	...	2.6
Oracle9*i* Database[1]	▪ SQL ▪ PL/SQL	Version that Oracle E-Business Suite certified to.
Oracle9*i*AS	OC4J	9i Release 3

[1] Installed with Oracle E-Business Suite.

See Also:

`http:\\otn.oracle.com/documentation/content.html`

2.1.2 Oracle E-Business Suite

HTB is a product of the Oracle E-Business Suite. To implement HTB, you must first install E-Business Suite 11*i* by performing the following steps:

1. Find the documents listed in Table 2–2, available from Oracle*MetaLink* (`http://metalink.oracle.com`); these documents describe how to install Oracle E-Business Suite:

Table 2–2 E-Business Suite Installation Documents on Oracle MetaLink

Document ID[1]	Document Title
66926.1	Oracle Applications 11i Info Center
132072.1	SST: Oracle Applications Installation and Upgrade
216550.1	Interoperability Notes

[1] Use Advanced Search on the Document ID.

2. Ensure that your Oracle Applications instance is at Release 11.5.7 *with the Release 11.5.7 mini-pack installed*—before proceeding with the installation of HTB patches.
3. Upgrade your database to Oracle9i Release 2.
4. Apply the HTB patches.

> **Note:** Patch numbers will be provided in a future release of this Implementation Guide.

2.1.3 Oracle Containers for J2EE

After completing the Oracle 11i installation, you must install Oracle9*i*AS *Containers for J2EE (OC4J),* by performing the following steps:

1. Download Oracle9*i*AS *Containers for J2EE* from the Oracle Technology Network web site (`http:\\otn.oracle.com`).
2. After downloading OC4J from Oracle Technet (OTN), unzip the file and record the directory location—referred to as `OC4J_HOME` for ease of reference in this document.
3. At the command line, navigate to your `OC4J_HOME`/j2ee/home directory and run the following command:

```
java -jar oc4j.jar -install
```

4. Enter the Administrator password.

> **Note:** The Administrator password is required to install the HTB EAR file (Enterprise EJBs).

2.1.4 HTB EAR File Installation

To install the HTB EAR file:

1. Change the current directory to `OC4J_HOME/j2ee/home/.`
2. Add the following lines in the library path section of `OC4J_HOME/j2ee/home/config/application.xml`:

   Note: *Replace `$JAVA_TOP` with the absolute path of your `$JAVA_TOP`.*

   ```
   <library path="../../../jdbc/lib/classes12dms.jar" />
   <library path="$JAVA_TOP/apps.zip" />
   ```

3. Start the OC4J Container with the following command:

   ```
   java -Xmx512M -Xms512M -jar oc4j.jar
   ```

   Note: *Memory sizes are approximated.*

4. A **message** confirms that the OC4J container has been initialized.
5. Locate the `htb.ear` file on `$APPL_TOP/ctb/11.5.0/java/ear/`.

   Note: *You will need the administrator password you created when installing the OC4J container.*

   Use the following command to install the `htb.ear` file:

   ```
   java -Xmx512M -Xms512M -jar admin.jar ormi://localhost
   admin <password> -deploy -file $APPL_TOP/ctb/11.5.0/java/ear/htb.ear
   -deploymentName htb
   ```

See Also: About Oracle Healthcare Family Pack D (Preface)

2.1.5 Configuring your HTB Installation

To configure your HTB installation:

1. You now have a directory with the path `OC4J_HOME/j2ee/home/application-deployments/htb`.

   Within this directory, edit the file `orion-application.xml`:

   - Find the section referencing a property `DBC_FILE_PATH`.
   - Change this value to correspond to the DBC file used by your Oracle11*i* instance. Remember this location—it will be referred to as `DBC_FILE_PATH`.

2. Restart the OC4J container using the following commands in your `OC4J/home/j2ee` directory:

   ```
   java -jar admin.jar ormi://localhost admin password -shutdown force
   ```

   Note: *Verify that the process is no longer running before restarting.*

   ```
   java -Xmx512M -Xms128M -Djbo.323.compatible=true -jar oc4j.jar
   ```

3. Your HTB installation is now complete.

See Also: About Oracle Healthcare Family Pack D (Preface)

2.1.6 Client Environment Library Requirements

Client-side libraries are required to develop and run client programs. You can access most of these libraries on the server, and download the balance from Oracle Technology Network.

Server-side libraries are listed by Table 2–3:

Table 2–3 Server-Side Libraries

Library File Name	Location (Directory Path)
classes12dms.jar	$OC4J_HOME/jdbc/lib
ejb.jar	$OC4J_HOME/j2ee/home/lib
htbclnt.jar	$APPL_TOP/ctb/11.5.0/java/jar
jaas.jar	$OC4J_HOME/j2ee/home/lib
jms.jar	$OC4J_HOME/j2ee/home/lib

Table 2–3 (Cont.) Server-Side Libraries

Library File Name	Location (Directory Path)
jndi.jar	$OC4J_HOME/j2ee/home/lib
jta.jar	$OC4J_HOME/j2ee/home/lib
saxon.jar	$OC4J_HOME/j2ee/home/lib
oc4jclient.jar	$OC4J_HOME/j2ee/home

To obtain the balance of the libraries, download Oracle JDeveloper 9.0.3 (`jdeveloper.zip`) from Oracle Technology Network (`http:\\otn.oracle.com`) and unzip the following files:

- bc4jct.jar
- bc4jdomorcl.jar
- bc4jmt.jar

Part II

Implementing Oracle Healthcare Transaction Base

This section of the Oracle Healthcare Transaction Base Implementation Guide describes the implementation process, and contains the following chapters:

- Chapter 3, Implementation Overview
- Chapter 4, Implementation Tasks: HTB Platform
- Chapter 5, Implementation Tasks: Payer Financial Management

3

Implementation Overview

This chapter describes the implementation process for the Oracle Healthcare Transaction Base platform, including the implementation task sequence.

This chapter contains the following topics:

- Section 3.1, Implementation Process Description
- Section 3.2, Implementation Tasks

See Also:

- Chapter 4, Implementation Tasks: HTB Platform, for a description of each procedure required to implement the HTB Platform.
- Chapter 5, Implementation Tasks: Payer Financial Management, for information about implementing Oracle Payer Financial Management.

3.1 Implementation Process Description

Oracle Healthcare Transaction Base provides two fundamental interfaces to implement its core services: (i) a **window user interface**, available for selected services, and (ii) the HTB Session Service, which provides an API call interface to all methods supported by HTB. Some implementation procedures incidentally employ a command line interface as well.

To complete the implementation of Oracle Healthcare Transaction Base, you must implement each core service listed by Table 3–1. This table cross-references implementation procedures applicable to each service, and is followed by individual procedure reference tables for each implementation procedure.

To Implement Services Using the Window User Interface:

[Window interface specified by applicable implementation reference table—tables 3–2 through 3–14]

1. Log in to Oracle Applications as `SYSADMIN`.

> **Caution: Although user SYSADMIN is assigned all of the seeded responsibilities, you should avoid routinely using this user name for daily administrative tasks—*create your own System Administrator user names for managing HTB security.***

2. Select the appropriate **responsibility**—indicated separately for each implementation procedure (and by its associated reference table—tables 3–2 through 3–11).
3. Navigate to the applicable window and perform the procedure.

To Implement Services Using the HTB Session Service:

[Session Service interface specified by applicable implementation reference table—tables 3–2 through 3–11]

1. Use the HTB Session Service to initiate a user session, as described by Appendix A, HTB Session Service.
2. Select the appropriate **role**—indicated separately for each implementation procedure (and by the its associated reference table—tables 3–2 through 3–11).
3. Perform the procedure.

See Also: *Oracle Javadoc for HTB*, for information about methods supported by the HTB platform.

3.2 Implementation Tasks

Figure 3–1 provides an overview of the implementation process for the HTB Platform—the tables that follow it (Tables 3–1 through 3–14) cross-reference all tasks by implementation procedure:

Figure 3–1 Implementation Task Process: HTB Platform

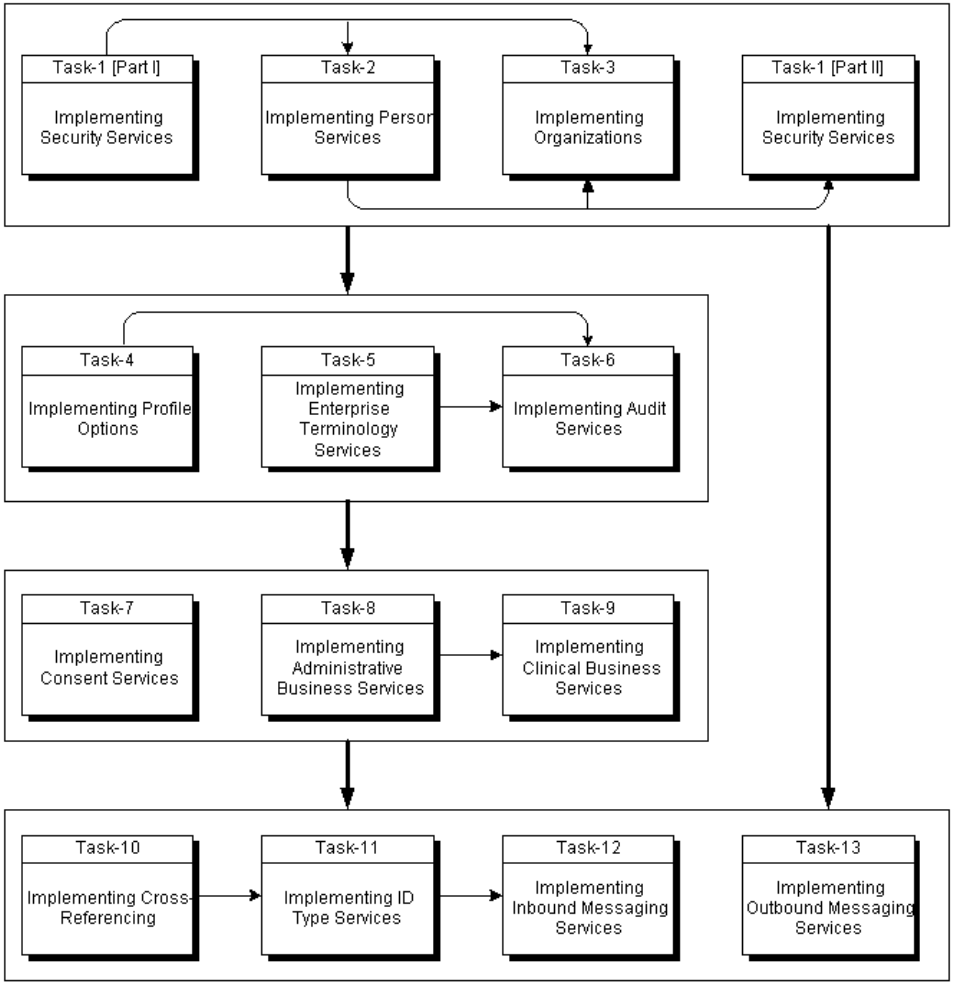


Figure 3–1 **Notes:**

- This chart indicates the implementation task sequence as well as dependency relationships between tasks.
- You can generally complete tasks sequentially, or in the sequence indicated by this chart. *But note that where a prerequisite is indicated, the predecessor task must be completed first.*
- You must start with Task-1 (completing Steps 1-1 through 1-9), after which you can perform tasks 2 and 3 in parallel.
- Following Task-3, you must complete Step 1-10 of Task-1 (*Implementing Organization-based Security*), after which you can proceed with Tasks 4 and 5 in parallel; both of these must be completed before performing Task-6.
- You can then perform tasks 7 and 8 in parallel, after which you can perform Task-9.
- You must complete Task-10 before performing Task-11, which in turn precedes Task-12.
- Perform Task-13 (following completion of Tasks 1 -3) to complete the HTB implementation process.
- Detailed implementation flow charts are included with each procedure section in Chapter 4—please note the *Prerequisites* defined for each procedure.

Table 3–1 summarizes the HTB implementation tasks and cross-references individual procedure reference tables for each implementation procedure (tables 3–2 through 3–11).

The Section Reference column of Table 3–1 cross-references the appropriate implementation procedure section for each task, and the Table Reference column cross-references the individual procedure tables that follow. The actual procedures are described by Chapter 4. You can perform these tasks in the sequence indicated by either Table 3–1 or by Figure 3–1:

Table 3–1 HTB Implementation Tasks Summarized

Task	Description	Yes[1] No	Section Reference	Table Reference
1	Implementing Security Services	Yes	Section 4.1	Table 3–2
2	Implementing Person Services	Yes	Section 4.2	Table 3–3
3	Implementing Organizations	Yes	Section 4.3	Table 3–4
4	Implementing Profile Option Services	No	Section 4.4	Table 3–5
5	Implementing Enterprise Terminology Services	Yes	Section 4.5	Table 3–6
6	Implementing Audit Services	No	Section 4.6	Table 3–7
7	Implementing Consent Services	Yes	Section 4.7	Table 3–8
8	Implementing Administrative Business Services	Yes	Section 4.8	Table 3–9
9	Implementing Clinical Business Services	Yes	Section 4.9	Table 3–10
10	Implementing Cross-Referencing	No	Section 4.10	Table 3–11
11	Implementing ID Type Services	No	Section 4.11	Table 3–12
12	Implementing Inbound Messaging Services	No	Section 4.12	Table 3–13
13	Implementing Outbound Messaging Services	No	Section 4.13	Table 3–14

[1] Required?

Table 3–2 HTB Implementation Procedures: Security Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
1-1	Configuring Password and Session Policies	Yes	Healthcare Security Administrator	Window	Section 4.1.1
1-2	Configuring Login URL for Notification Messages	Yes	Healthcare Security Administrator	Windows	Section 4.1.2
1-3	Creating Accounts	Yes	Healthcare Security Administrator	Window	Section 4.1.3
1-4	Creating Responsibilities	No	Healthcare Security Administrator	Session Service	Section 4.1.4
1-5	Assigning Accounts to Responsibilities	No	Healthcare Security Administrator	Window	Section 4.1.5
1-6	Creating Security Criteria on Protected Objects	Yes	Healthcare Security Administrator	Window	Section 4.1.6
1-7	Creating Protected Functions	No	Healthcare Security Administrator	Session Service	Section 4.1.7

Table 3–2 (Cont.) HTB Implementation Procedures: Security Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
1-8	Assigning Functions to Responsibilities Conditioned by Criteria	No	Healthcare Security Administrator	Window	Section 4.1.8
1-9	Implementing Break-The-Glass	No	Healthcare Security Administrator	Session Service	Section 4.1.9
1-10	Implementing Organization Based Security	No	Healthcare Security Administrator	Window or Session Service	Section 4.1.10

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–3 HTB Implementation Procedures: Person Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
2-1	Defining Person Services Profile Options for DQM	Yes	Healthcare Configuration Administrator	Window	Section 4.2.1
2-2	Adding Trading Community Manager Responsibility	Yes	System Administrator	Window	Section 4.2.2
2-3	Implementing TCA DQM for HTB	Yes	Trading Community Manager	Window	Section 4.2.3
2-4	Implementing Person Domain Service	No	User-defined	Session Service	Section 4.2.4
2-5	Implementing Person Management	Yes	User-defined	Session Service	Section 4.2.5
2-6	Implementing Correlation	Yes	User-defined	Session Service	Section 4.2.6

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–4 HTB Implementation Procedures: Organizations

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
3-1	Implementing Organization Units	Yes	User-Defined Responsibility	Session Service	Section 4.3.1
3-2	Implementing Organization Roles	Yes	User-Defined Responsibility	Session Service	Section 4.3.2
3-3	Implementing Organization Hierarchies	No	User-Defined Responsibility	Session Service	Section 4.3.3

[1] Required?

[2] There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–5 HTB Implementation Procedures: Profile Option Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
4-1	Implementing Profile Options	No	Healthcare Application Developer	Window	Section 4.4.1
4-2	Implementing Profile Option Values	Yes	Healthcare Configuration Administrator	Window	Section 4.4.2

[1] Required?

[2] There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–6 HTB Implementation Procedures: Enterprise Terminology Services (ETS)

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
5-1	Creating New Generic Coding Schemes	Yes	Healthcare ETS Administrator	Window	Section 4.5.1
5-2	Loading and Activating New Terminology Versions	No	Healthcare ETS Administrator	Window/ Command Line	Section 4.5.2
5-3	Creating and Loading ETS Cross Maps	No	Healthcare ETS Administrator	Command Line/ Window	Section 4.5.3
5-4	Creating Local Descriptions	No	Healthcare ETS Administrator	Window	Section 4.5.4

Table 3–6 (Cont.) HTB Implementation Procedures: Enterprise Terminology Services (ETS)

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
5-5	Managing Usage Contexts	No	Healthcare ETS Administrator	Window	Section 4.5.5
5-6	Implementing Concept Lists	No	Healthcare ETS Administrator	Window	Section 4.5.6
5-7	Implementing Interterminology and Intraterminology Equivalence	No	Healthcare ETS Administrator	Window / Command Line / Session Service	Section 4.5.7
5-8	Other ETS Support Functions	No	Healthcare ETS Administrator	Window	Section 4.5.8

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–7 HTB Implementation Procedures: Audit Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
6-1	Enabling Audit Services	No	Healthcare Configuration Administrator	Window	Section 4.6.1
6-2	Initializing Existing Audit Event Types	No	Healthcare Configuration Administrator	Window	Section 4.6.2
6-3	Creating New Audit Event Types	Yes	Healthcare Application Developer	Window	Section 4.6.3
6-4	Invoking HTB Audit Services	No	User-Defined	Session Service	Section 4.6.4
6-5	Attribute Values in Audit Events	Yes	Healthcare Application Developer; Healthcare ETS Administrator	Window	Section 4.6.5

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–8 Implementation Procedures: Patient Authorization and Consent Services

Task Step	Description	Yes/ No[1]	Performed by (Responsibility)	Interface[2]	Procedure Reference
7-1	Implementing Patient Authorization	Yes	User Defined	Session Service	Section 4.7.1
7-2	Implementing Patient Consent	Yes	User Defined	Session Service	Section 4.7.2
7-3	Implementing Patient Opt-Out	Yes	User Defined	Session Service	Section 4.7.3

[1] Required?

[2] There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–9 HTB Implementation Procedures: Administrative Business Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
8-1	Implementing Payers and Plans	Yes	User-Defined Responsibility	Session Service	Section 4.8.1
8-2	Implementing Employers	Yes	User-Defined Responsibility	Session Service	Section 4.8.2
8-3	Implementing Address Management	Yes	User-Defined Responsibility	Session Service	Section 4.8.3
8-4	Implementing Staff Management	Yes	User-Defined Responsibility	Session Service	Section 4.8.4
8-5	Implementing Care Sites	Yes	User-Defined Responsibility	Session Service	Section 4.8.5
8-6	Implementing Equipment Management	Yes	User-Defined Responsibility	Session Service	Section 4.8.6
8-7	Implementing Resource Management	No	User-Defined Responsibility	Session Service	Section 4.8.7
8-8	Implementing Workgroups	Yes	User-Defined Responsibility	Session Service	Section 4.8.8
8-9	Implementing Patient Registration	No	User-Defined Responsibility	Session Service	Section 4.8.9
8-10	Implementing Wait List Management	No	User-Defined Responsibility	Session Service	Section 4.8.10

Table 3–9 (Cont.) HTB Implementation Procedures: Administrative Business Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
8-11	Implementing Patient Scheduling	No	User-Defined Responsibility	Session Service	Section 4.8.11
8-12	Implementing Encounter Management	Yes	User-Defined Responsibility	Session Service	Section 4.8.12
8-13	Implementing Patient List Management	No	User-Defined Responsibility	Session Service	Section 4.8.13

[1] Required?

[2] There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–10 HTB Implementation Procedures: Clinical Business Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
9-1	Implementing Master Catalogs	Yes	User-Defined Responsibility	Session Service	Section 4.9.1,
9-2	Implementing Clinical Act Service	Yes	User-Defined Responsibility	Session Service	Section 4.9.2
9-3	Implementing User Catalogs	Yes	User-Defined Responsibility	Session Service	Section 4.9.3
9-4	Implementing Clinical Applications	Yes	User-Defined Responsibility	Session Service	Section 4.9.4
9-5	Implementing a Medication Order Entry System	No	User-Defined Responsibility	Session Service	Section 4.9.5
9-4	Implementing a Results Reviewing System	No	User-Defined Responsibility	Session Service	Section 4.9.6

[1] Required?

[2] There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–11 HTB Implementation Procedures: Cross-Referencing

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
10-1	Implementing External Roots	No	User-Defined Responsibility	Session Service	Section 4.10.1
10-2	Implementing Cross-References	No	User-Defined Responsibility	Session Service	Section 4.10.2

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–12 HTB Implementation Procedures: ID Type Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
11-1	Creating the ID Type	**Yes**	User-Defined Responsibility	Session Service	Section 4.11.1
11-2	Updating ID Type	**No**	User-Defined Responsibility	Session Service	Section 4.11.2
11-3	Deleting ID Type	**No**	User-Defined Responsibility	Session Service	Section 4.11.3
11-4	Retrieving ID Types	**No**	User-Defined Responsibility	Session Service	Section 4.11.4

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–13 HTB Implementation Procedures: Inbound Messaging Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
12-1	Creating an IMP Administrator Responsibility and Assigning it to an Account	Yes	Healthcare Security Administrator	Window	Section 4.12.1
12-2	Assigning IMP Configuration Functions to the IMP Administrator Responsibility	Yes	Healthcare Security Administrator	Window	Section 4.12.2

Table 3–13 (Cont.) HTB Implementation Procedures: Inbound Messaging Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
12-3	Implementing ID Configuration	Yes	User-Defined Responsibility	Session Service	Section 4.12.3
12-4	Implementing Senders, Receivers, Trigger Events and Side Effects	No	User-Defined Responsibility	Session Service	Section 4.12.4
12-5	Invoking Inbound Messaging Services	Yes	User-Defined Responsibility	Session Service	Section 4.12.5

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

Table 3–14 HTB Implementation Procedures: Outbound Messaging Services

Task-Step	Description	Yes[1] No	Performed By (Responsibility)	Interface[2]	Procedure Reference
13-1	Creating and Assigning an OMP Administrator Responsibility	Yes	Healthcare Security Administrator	Window	Section 4.13.1
13-2	Assigning OMP Configuration Functions	Yes	Healthcare Security Administrator	Window	Section 4.13.2
13-3	Configuring OMP	Yes	Healthcare Security Administrator	Window, Session Service	Section 4.13.3
13-4	Invoking Outbound Messaging Services	Yes	User-Defined Responsibility	Session Service	Section 4.13.4
13-5	OMP Security	No	Healthcare Security Administrator	Session Service	Section 4.13.5
13-6	OMP Audit	No	Healthcare Security Administrator	Window	Section 4.13.6

1 Required?

2 There are three user interfaces used for selected implementation procedures: (i) a window interface; (ii) the HTB Session Service, which provides a call interface to HTB APIs; and (iii) a command line interface. See Also: Section 3.1, Implementation Process Description, and Appendix A, HTB Session Service.

See Also:

- Chapter 4, Implementation Tasks: HTB Platform
- Appendix A, HTB Session Service

4

Implementation Tasks: HTB Platform

This chapter describes implementation procedures for the HTB Platform, organized in implementation sequence.

This chapter contains the following topics:

> **See Also:** Chapter 3, Implementation Overview, for an overview of the implementation process, including the implementation task sequence and a description of the user interface.

4.1 Implementing Security Services

HTB Security Services authenticates users and lets security administrators define an **authorization** policy that specifies which users can invoke which APIs against which HTB objects.

Security administrators and administrative users can use HTB Security Manager, a window-based tool, to define authorization policy, including the following components:

- Accounts
- Organization Context (called Security Profiles in Oracle HRMS)
- Responsibilities (applicable to specific organizations)
- Assignment of accounts to responsibilities (applicable to specific organizations)
- Security criteria associated with protected objects
- Assignment of functions to responsibilities conditioned by criteria

For example, an authorization policy could state that only staff members in the Psychiatry department at Stanford Hospital who are assigned to a patient's care can view encounter data for that patient. This policy includes the following principal components (Table 4–1):

Table 4–1 Sample Security Policy

Account	Responsibility	Organization Context	Protected Function	Protected Object	Security Criteria
Dr Patel	`Psychiatry Staff`	Stanford Hospital	View `Encounters`	`Encounter`	User assigned to patient's care

Reference

- *Oracle Applications System Administrator's Guide*
- *Oracle Applications Installation Guide*

Prerequisites

- Implementing Person Services (required before proceeding with implementation Step 1-13; See: Figure 4-2):
  - TCA DQM must be implemented for HTB.
  - Person Services profile options must be set.

- Implementing Organizations (required before proceeding with implementation Step 1-13; See: Figure 4-2).

  See Also:

  - Section 4.2, Implementing Person Services
  - Section 4.2.3, Implementing TCA DQM for HTB
  - Section 4.3, Implementing Organizations
  - Section 4.4, Implementing Profile Option Services
  - Table D–1, Profile Options (Appendix D)

Login

Log in to Oracle Applications as `SYSADMIN`.

> **Caution: Although user SYSADMIN is assigned all of the seeded responsibilities, you should avoid routinely using this user name for daily administrative tasks—*create your own System Administrator user names for managing HTB security. If you create your own System Administrator user names, these users will only have access to Oracle Self Service Applications.***

Procedures

The implementation process for Security Services is organized in the following sequence:

- Part I: Performing Steps 1-1 through 1-12 (See: Figure 4–1)
- Part II:
  - Implementing Person Services
  - Implementing Organizations
  - Performing Step 1-13 (See: Figure 4–2)

Figure 4–1 Implementation Process: Security Services [Part-I]

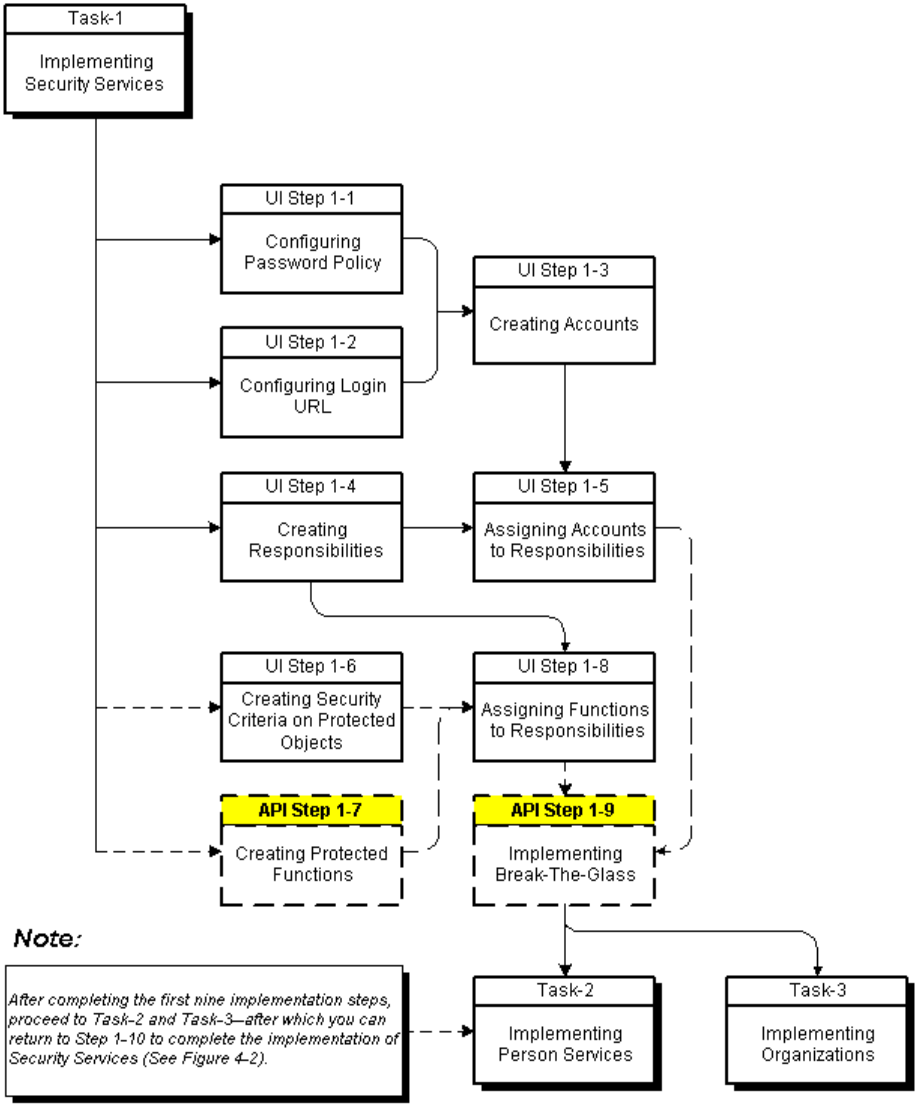
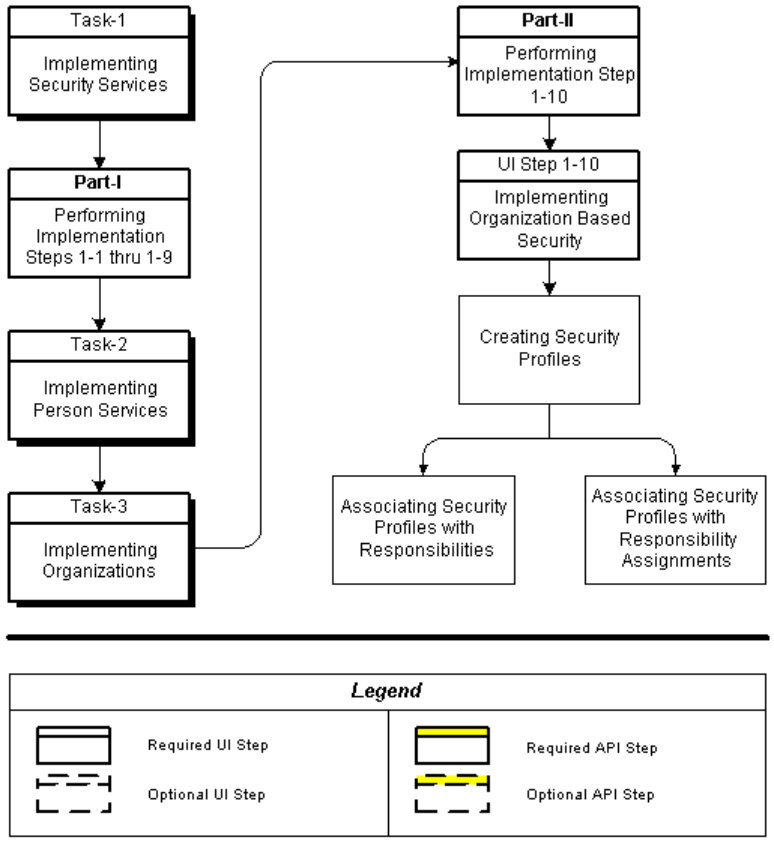


Figure 4–2 Implementation Process: Security Services [Part-II]



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–2, HTB Implementation Procedures: Security Services

The following sections describe the implementation procedures for Security Services (referenced by Figure 4–1):

- Section 4.1.1, Configuring Password and Session Policies
- Section 4.1.2, Configuring Login URL for Notification Messages
- Section 4.1.3, Creating Accounts
- Section 4.1.4, Creating Responsibilities
- Section 4.1.5, Assigning Accounts to Responsibilities
- Section 4.1.6, Creating Security Criteria on Protected Objects
- Section 4.1.7, Creating Protected Functions
- Section 4.1.8, Assigning Functions to Responsibilities Conditioned by Criteria
- Section 4.1.9, Implementing Break-The-Glass
- Section 4.1.10, Implementing Organization Based Security

4.1.1 Configuring Password and Session Policies

You can configure password and session policies by setting profile options.

4.1.1.1 Password Structure

To configure password structure:

- Set the E-Business Suite profile option *Signon Password Length* to specify minimum required password length in characters. The default is six characters.
- Set the E-Business Suite profile option *Signon Password Hard to Guess* profile to specify required password complexity. If set to *yes*, this profile option enforces the following password rules:
  - Passwords must contain at least one letter and at least one digit.
  - Passwords cannot contain user names.
  - Passwords cannot contain repeating characters.

Responsibility

System Administrator

4.1.1.2 Password Expiration

To configure password expiration:

Set the HTB profile option `CTB_SIGNON_PASSWORD_EXPIRATION_LIMIT` profile option at the Site level to specify password expiration (at time of account creation) in terms of the maximum number of days between password changes; passwords expire at the conclusion of the number of days specified, starting with the day following account creation date. By default, passwords never expire. Changes to the value of the profile option only apply to new accounts created after the change.

Responsibility

Healthcare Configuration Administrator

Navigation

Table 4–2 summarizes the navigation paths used by this section:

Table 4–2 Navigation Paths: Configuring Password Expiration

Function or Window	Navigation Path
System Profile Option	`System Profile Option Values > View Profile Option Values > Update`

4.1.1.3 Session Policy

To configure session timeout and account lockout:

- Session timeout can be configured through three E-Business Suite profile options:
  - **ICX: Session Timeout:** The amount of time between session validations. A session is validated each time an HTB API is called. The default is no inactivity timeout.
  - **ICX: Limit time:** The total amount of time in hours from the session creation time. Defines the amount of time that a session is valid. The default is four hours.
  - **ICX: Limit Connect:** The number of times a session is validated. A session is validated each time an HTB API is called. The default is 1,000.
- **Account Lockout:** You can set the E-Business Suite profile option *Signon Password Failure Limit* to specify the maximum number of unsuccessful login attempts, after which the session is terminated and the account is locked out (end-dated).

Responsibility

System Administrator

See Also: For more information about setting profile options:

- *Oracle Applications System Administrator's Guide, Setting User Profile Options*
- *Oracle Self-Service Web Applications Implementation Guide*

4.1.2 Configuring Login URL for Notification Messages

When an account password is reset, a message is sent to the account e-mail address notifying the account owner of the password change. Included in the message is a URL through which the account owner can log in to HTB-based applications.

For example, several HTB-based applications can be accessible from a centralized portal page. The URL of the portal page can be included in password change e-mails.

The value of this login URL is a profile option that can be configured at install time, using the HTB Configuration Manager GUI.

Responsibility

Healthcare Configuration Administrator

Navigation

Table 4–3 summarizes the navigation paths used by this section:

Table 4–3 Navigation Paths: Configuring Login URL

Function or Window	Navigation Path
Create Profile Option Value	`System Profile Option Values > Profile Options > View Profile Option Values > Create Profile Option Value`

Steps

1. Navigate to the Profile Options page and search for `CTB: Login URL.`
2. Select the profile option and navigate to the View Profile Option Values page.
3. Navigate to the Create Profile Option Values page and select `level = Site.`
4. Enter the URL and click Apply.

4.1.3 Creating Accounts

An active entity must establish its rights to access HTB. An account represents either a human user or a system entity that is registered in and authentic to HTB. Examples of system entities include a root system account for system administrators, and a virtual account used by HTB messaging services to log in to HTB.

> **Note:** Not every person known to a healthcare information system is a user. Patients, for example, are typically not HTB users.

An account has the following attributes:

- Login ID: Identifier for the account
- Password
- Effective date and expiration date: Indicates whether the account is active
- E-mail address: Used to send account change notifications

For accounts that represent human users, an additional attribute, PersonID, identifies the Person object in HTB Person Service that represents the user.

Login

Log in to Oracle Applications as an administrative user.

Responsibility

Healthcare Security Administrator

Navigation

Table 4–4 summarizes the navigation paths used by this section:

Table 4–4 Navigation Paths: Creating Accounts

Function or Window	Navigation Path
Create Person	`Persons > Create Person`
Create System Account	`Accounts > Create System Account`
Create User Account	`Accounts > Create User Account`

Steps:

> **Note:** *Implementing Person Services is a prerequisite to creating a user account directly (step 1)*. Alternatively, you can create a system account first (Step 2), and associate a person with that account after implementing Person Services.

1. If the account represents a person:
   - Navigate to the Persons Search page (Simple or Advanced) and search for the person; if the person does not exist, navigate to the Create Person page to create the person.
   - Navigate to the Create User Account page to create an account for the person.
2. If the account does not represent a person, navigate to the Create System Account page to create a system account.
3. When an account is successfully created, a password is automatically generated and sent to the e-mail address of the account through Oracle Workflow. To receive password notification e-mail, you must configure Oracle Workflow with the global notification preference set to *e-mail*.

   See Also: *Oracle Workflow Administrator's Guide*, Chapter 2, for information about setting up and configuring Oracle Workflow.

4.1.4 Creating Responsibilities

A **responsibility** represents a job function that has associated permissions to access HTB. Responsibilities group users with similar job functions. Examples include Clinician, Admissions Clerk, Nurse...

Login

Log in to Oracle Applications as an administrative user.

Responsibility

Healthcare Security Administrator

Navigation

Table 4–5 summarizes the navigation paths used by this section:

Table 4–5 Navigation Paths: Creating Responsibilities

Function or Window	Navigation Path
Add Responsibility	`Responsibility > Add Responsibility`

Steps

1. Navigate to the Responsibility Search Page to search for the responsibility.
2. Navigate to the Create Responsibility page to create the responsibility.

Default

Table 4–6 lists the default responsibilities created at install time:

Table 4–6 Seeded Responsibilities

Responsibility Name	Description
Healthcare Security Administrator	`Users of this responsibility administer accounts and security policies.`
Healthcare Configuration Administrator	`Users of this responsibility administer profile options.`
Inbound Message Processor	`Used by Inbound Message Processor to process inbound messages.`
Outbound Message Processor	`Used by Outbound Message Processor to process outbound messages.`

4.1.5 Assigning Accounts to Responsibilities

An account can be assigned to one or more responsibilities—which indicate that the human user or system entity associated with the account is authorized to perform the job functions represented by the responsibilities.

Login

Log in to Oracle Applications as an administrative user.

Responsibility

Healthcare Security Administrator

Navigation

Table 4–7 summarizes the navigation paths used by this section:

Table 4–7 Navigation Paths: Assigning Accounts to Responsibilities

Function or Window	Navigation Path
Assign Account to Responsibilitys	`Accounts > View Assigned Responsibilitys > Assign Account to Responsibilitys`
Assign Accounts to Responsibility	`Responsibilitys > Responsibilities Search > View Responsibility Details > Assign Accounts to Responsibility`

Steps:

You can search for the *user or system account*, or you can search for the *responsibility*:

- To search for the *user or system account:*
  1. Navigate to the Accounts Search page (Simple or Advanced).
  2. Select the account.
  3. Navigate to the View Assigned Responsibilitys page.
  4. Navigate to the Assign Account to Responsibilities page.
  5. Click the torchlight to navigate to the Search and Select page.
  6. Enter a partial responsibility name and click Go.
  7. Click the responsibility name to be selected.
  8. Return to step 5 as necessary.
- To search for the *responsibility*:
  1. Navigate to the Responsibilities Search page.
  2. Click the responsibility name to navigate to the View Responsibility Detail page.
  3. Navigate to the Assign Accounts to Responsibility page.
  4. Click the torchlight to navigate to the Search and Select page.
  5. Enter a partial account name and click Go.
  6. Click the account name to be selected.
  7. Return to step 4 as necessary.

Seeded Responsibility Assignments

Table 4–8 lists seeded responsibility assignments:

Table 4–8 Seeded Users and Responsibilities

Users	Responsibilities
SYSADMIN	■ Healthcare Configuration Administrator ■ Healthcare Security Administrator

4.1.6 Creating Security Criteria on Protected Objects

4.1.6.1 Protected Objects

Protected **object**s are HTB objects queried or manipulated by protected functions. These objects are related to each other and to other HTB objects through object reference attributes.

For example, encounter objects have an attribute `PatientIdentifier,` which references the associated Patient object. The following charts (Figure 4–3 through Figure 4–7) show how protected objects are related through object references:

Figure 4–3 Protected Objects: Person and Patient Services

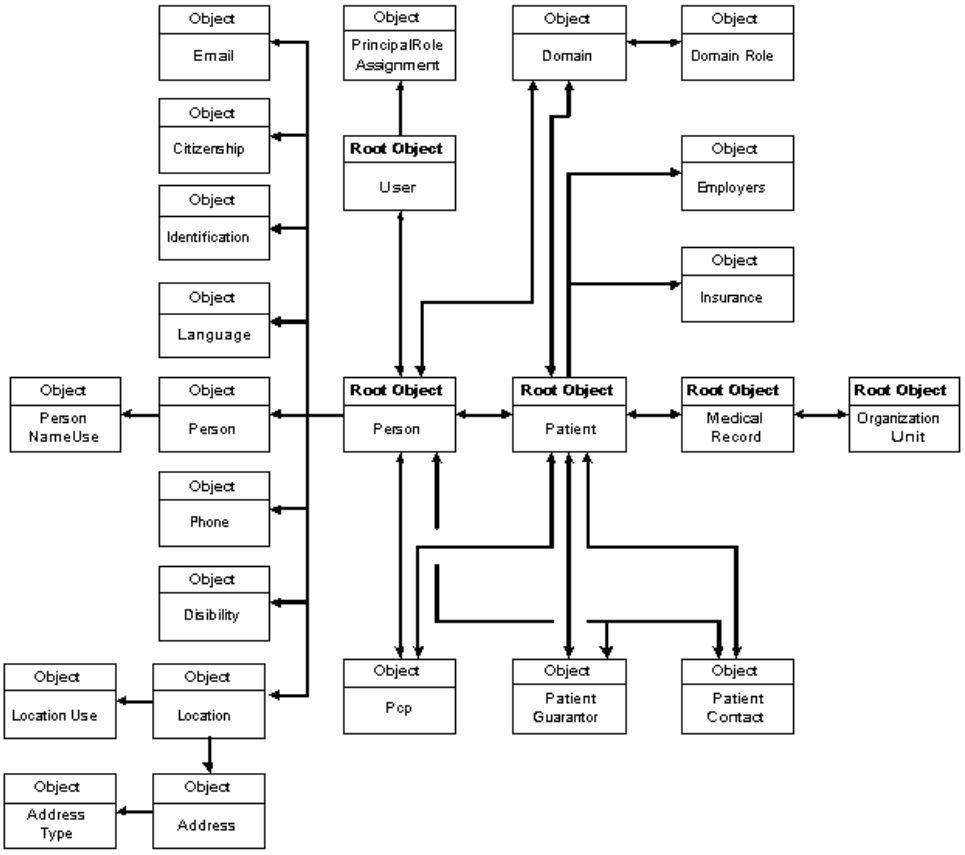


Figure 4–4 Protected Objects: Organization and Staff Services

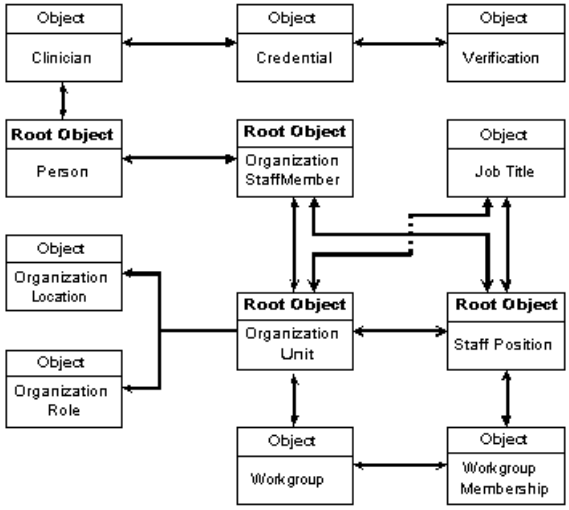


Figure 4–5 Protected Objects: Encounter Service

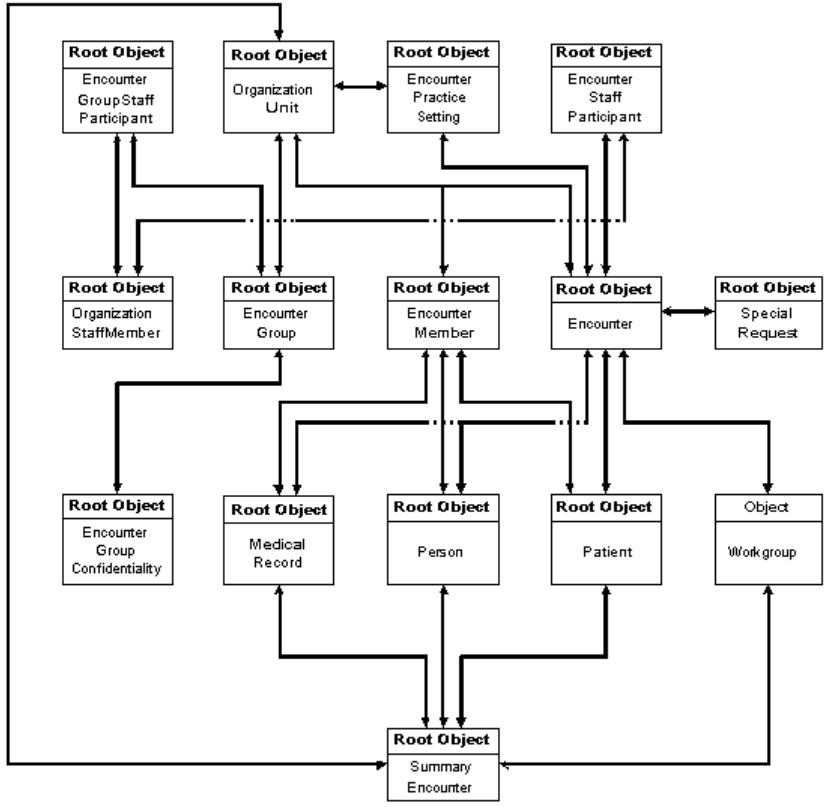


Figure 4–6 Protected Objects: ClinicalAct Service

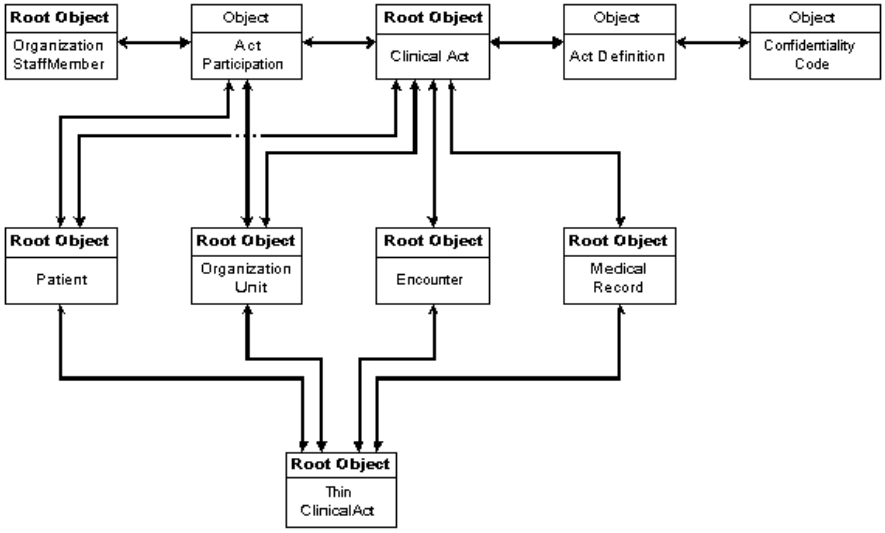
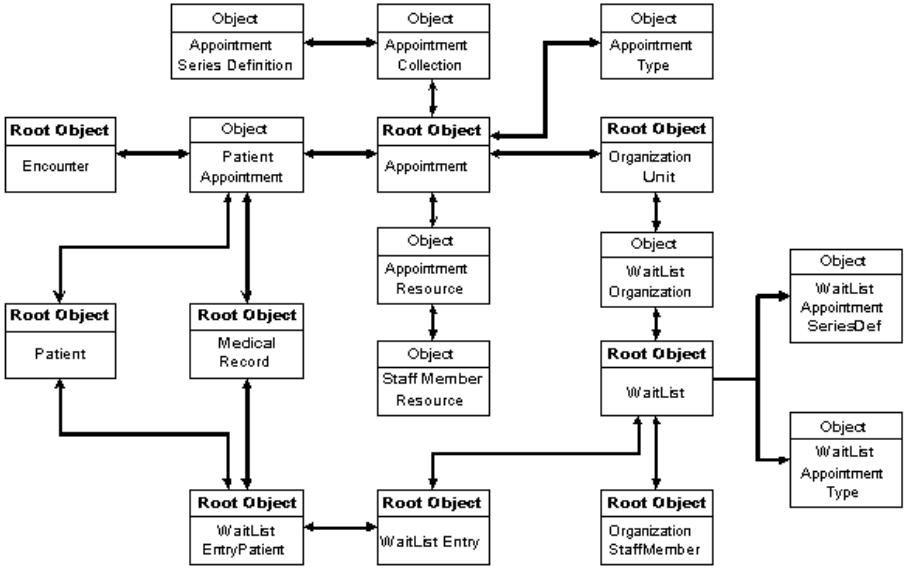


Figure 4–7 Protected Objects: Scheduling and Wait List Services



4.1.6.2 Security Criteria

Security criteria are constraints attached to protected objects, which define a subset of protected objects. The constraints are applied to attributes of the protected objects.

For example, security criteria for encounter objects can be defined using the `ConfidentialityCode` attribute with the statement in Example 4–1:

Example 4–1 Using ConfidentialityCode Attribute

```
ConfidentialityCode != "PSY"
```

This defines the subset of non-psychiatry encounters.

The constraint can also be applied to attributes of objects related to the protected objects displayed by Figure 4–3 through Figure 4–7.

For example, a security criterion might be applied to encounter objects with the statement in Example 4–2:

Example 4–2 Using Security Criterion

```
VipCode != "VIP"
```

where `VIPCode` is an attribute of `MedicalRecord` objects. This defines the encounter subset of VIP medical records.

In addition to comparing an attribute with a constant, security criteria can also be defined to compare an object attribute with an attribute in the session context. HTB supports the following imbedded context attribute:

`LoginPersonId:` The identifier of the Person object that corresponds to the logged in user. This attribute is automatically derived after authentication, and cannot be set by applications.

Example 4–3 LoginPersonId:

```
PersonIdentifier = :LoginPersonId
```

where `PersonIdentifier` is an attribute of encounter objects. It defines the subset of encounters for which the logged-in user is the patient.

Login

Log in to Oracle Applications as an administrative user.

Responsibility

Healthcare Security Administrator

Navigation

Table 4–9 summarizes the navigation paths used by this section:

Table 4–9 Navigation Paths: Security Criteria

Function or Window	Navigation Path
Add Child	`Criteria > Criteria Search > View Criteria Details > Add Child`
Create Criteria	`Criteria > Create Criteria`
Update Criteria Statement	`Criteria > Criteria Search > View Criteria Details > Update Criteria Statement`

Steps:

You can create new criteria or update existing criteria.

- To create criteria:
  1. Navigate to the Create Criteria page.
  2. Enter the criteria name, select the root object, and click Apply.
- To update existing criteria:
  1. Navigate to the Criteria Search page and search for the criteria.
  2. Click the criteria name.
  3. Select a criteria node as the current object:
     - Navigate to the Update Criteria Statement page to add a constraint on an attribute of the current object, *or,*
     - Navigate to the Add Child page to add an object related to the current object in the protected object graphs.
  4. Repeat these steps as necessary.

Note: When adding a constraint on an attribute whose value is derived from a concept list, the concept list must be populated; it cannot be empty.

See Also: Section 4.5.6 for information about adding values to a concept list.

4.1.6.3 Upgrading Metadata Objects

During implementation of each new release of Oracle Healthcare Transaction Base, certain seeded metadata objects are added, updated, or deleted. As a consequence, some of the existing security objects are invalidated. New criteria and responsibilities cannot be defined using these invalid objects.

Accordingly, the System Administrator must perform the following steps to remove invalid responsibility assignments and invalid criteria:

- Report Invalidated Security Objects
- Delete Invalidated Responsibility Assignments
- Delete Invalidated Criteria

4.1.6.3.1 Reporting Invalidated Security Objects

A program named *CTB Security Upgrade Tool* is shipped with HTB. This program generates a report that lists security objects that have been invalidated by the upgrade, including (i) invalid criteria and (ii) invalid responsibilities. Prior to running this report you must assign the *CTB Security Upgrade Report* to *System Administrato*r responsibility. You can then run it using the Oracle Concurrent Manager.

Steps

Step 1: Assigning System Administrator Responsibility

1. Log in as System Administrator (use the account name you created in Section 4.1.3).
2. Select System Administrator responsibility.
3. Select *Security* > *Responsibility* > *Request* from the main menu.
4. Select *System Administrator Reports* request group. This is the request group associated with the System Administrator responsibility.
5. In the Requests panel, select the following values—one for each of the three columns displayed:
   - Type Column: *Select Program.*
   - Name Column: *Select CTB Security Upgrade Report*
   - Application Column: *Select Oracle Healthcare Transaction Base*
6. Save the changes.

Step 2: Running the Concurrent Program

1. Log in as System Administrator (use the account name you created in Section 4.1.3).
2. Select System Administrator responsibility.
3. Select *Request* > *Run* from the main menu.
4. Click OK.
5. Enter the concurrent program name: *CTB Security Upgrade Report*
6. Click OK and Submit, in turn.

7. After the request is completed, click View Output to see the report; the report lists invalid criteria and responsibilities.
8. Click View Log to see the Log File entries.

4.1.6.3.2 Deleting Invalid Responsibility Assignments

Steps

1. Change your responsibility to Healthcare Security Administrator.
2. Navigate to the Responsibility Search page; click *Responsibilities* > *Search* for the invalid responsibilities.
3. Delete the invalid assignments marked with *(X)* by clicking the Unassign Function icon.
4. Repeat this process for all invalid responsibilities.

4.1.6.3.3 Deleting Invalid Criteria

Steps

1. Navigate to the Criteria Search page (click the Criteria tab); search for all invalid criteria.
2. Click the Delete Criteria icon.
3. Repeat this process for all invalid criteria listed in the report.

4.1.7 Creating Protected Functions

A protected function represents an HTB API requiring prior authorization. Most HTB service APIs are represented as protected functions.

For example, Find Encounters is a protected function that represents the `findEncounters` method in HTB Encounter Service. A user cannot invoke this method without authorization.

In addition to protected functions that represent HTB service APIs, HTB-based applications can create or delete application-specific protected functions through the following methods:

- `createFunctions`
- `deleteFunctions`
- `getFunction`

Reference

Oracle Javadoc for HTB

Table 4–10 lists the principal methods that relate to protected functions:

Table 4–10 Service and Methods: Protected Functions

Level	Detail
Package	`oracle.apps.ctb.security`
Class	`SecurityManagementService`
Methods	▪ `createFunctions` ▪ `deleteFunctions` ▪ `getFunction`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibilities

- Healthcare Security Administrator for setup
- User-defined responsibility to execute

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Default

Protected functions that represent HTB service APIs are created at install time.

4.1.8 Assigning Functions to Responsibilities Conditioned by Criteria

Protected functions can be assigned to responsibilities to indicate that accounts assigned to such responsibilities are authorized to invoke APIs represented by the protected functions.

For example, the Find Encounters function can be assigned to the Clerk responsibility, which authorizes clerks to view encounters.

When a protected function represents an API that queries or updates HTB objects, the assignment of the function to a responsibility could optionally be conditioned by a security criterion rooted on HTB objects. It could state that accounts assigned to the responsibility are authorized to invoke the API to query or update HTB objects that satisfy the criterion.

For example, assume that you have created criteria on encounter objects with the constraint in Example 4–4:

Example 4–4 Security Criterion

```
ConfidentialityCode != "PSY"
```

When you assign the function Find Encounters to the Clerk responsibility conditioned by this criteria, you are authorizing clerks to view non-psychiatry encounters.

Login

Log in to Oracle Applications as an administrative user.

Responsibility

Healthcare Security Administrator

Navigation

Table 4–11 summarizes the navigation paths used by this section:

Table 4–11 Navigation Paths: Assigning Functions to Conditioned Responsibilitys

Function or Window	Navigation Path
Assign Functions to Responsibility	`Responsibilitys > Responsibility Search > View Responsibility Details > Assign Functions to Responsibility`

Steps

1. Navigate to the Responsibility Search page to find the responsibility.
2. Click the responsibility name to navigate to the View Responsibility Details page.
3. Navigate to the Assign Functions to Responsibility page to grant a function to the responsibility, optionally with a criterion.

Default

Table 4–12 defines assignments created at install time:

Table 4–12 Default Function Assignments

Responsibility Name	Functions
Healthcare Security Administrator	▪ CTB_SA_BREAK_THE_GLASS ▪ CTB_SA_RESET_PASSWORD
Inbound Message Processor	▪ CTB_MS_PROCESS_IN_MSG

4.1.9 Implementing Break-The-Glass

On an exception basis, authorization constraints can be turned off by *breaking the glass*. To break the glass, use the `SessionContext.setBreakTheGlass` method in the HTB Session Service. This method sets an attribute in the session context indicating if authorization is turned off.

When break-the-glass is turned on, authorization checks are bypassed for the rest of the session; calls to protected functions are not validated against security policies.

Not every user is authorized to break the glass. The `setBreakTheGlass` method is represented as a protected function. A user must be granted this function to invoke the method.

Reference

Oracle Javadoc for HTB

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibilities

- Healthcare Security Administrator for setup
- User-defined responsibility to execute

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

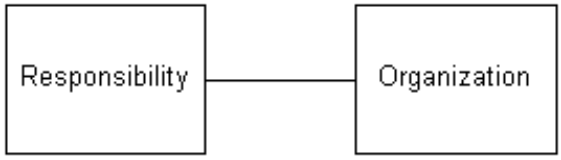
1. Create a responsibility (Section 4.1.4).
2. Assign the BreakTheGlass function to the responsibility (Section 4.1.8).
3. Assign applicable accounts to this responsibility (Section 4.1.5).

4.1.10 Implementing Organization Based Security

Session access rights can be affected by an organization context, which can be used to constrain access to value object subsets belonging to organizations in the context. There are two ways that an organization context can be defined:

- An organization context can be associated with a responsibility. Such a responsibility represents an organization-specific job function. In this model (Figure 4–8), there is a one-to-one mapping between the responsibility and the organization context. For example, a responsibility called *StanfordNurse* can be defined and associated with Stanford Hospital. At the beginning of the login session, a user selecting this responsibility is automatically associated with Stanford Hospital—the organization context for the session.

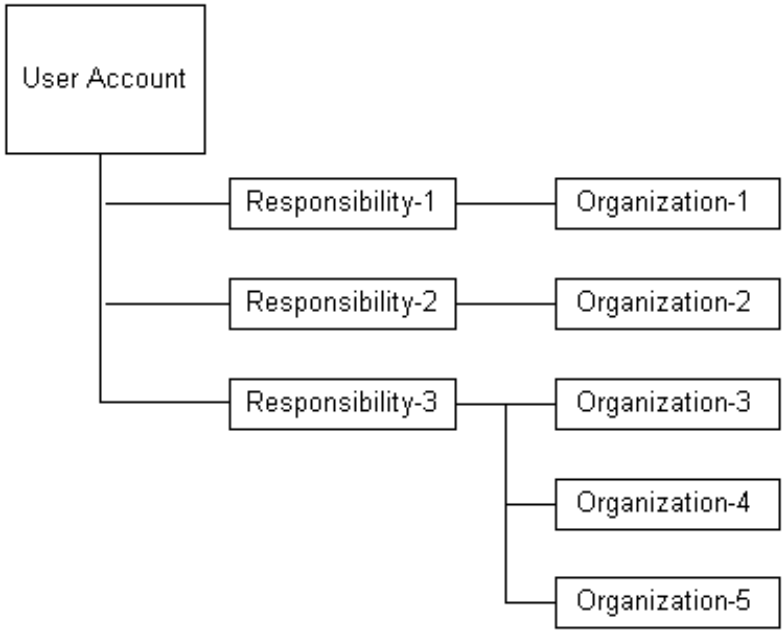
Figure 4–8 Organization-based Security: Single Organization



- An organization context can be associated with a user-responsibility assignment. The responsibility represents a job function shared among several organizations, and a user is assigned the job function at some but not necessarily all organizations. In this model (Figure 4–9), there is a three-way association between the user, responsibility and its associated organization contexts. For example, the responsibility *Nurse* could be shared among Stanford Hospital, Lucile Packard Children's Hospital, and UCSF. User Mary Smith is a nurse at Stanford Hospital and Lucile Packard Children's Hospital, but not UCSF. When she logs in and selects the Nurse responsibility, she can select

either Stanford Hospital or Lucile Packard Children's Hospital as the organization context of the session, but not UCSF.

Figure 4–9 Organization-based Security: Multiple Organizations



An organization context is represented as a security profile in HTB. It consists of one or more organization units in the organization hierarchy. When a session is constrained by an organization context, the value objects accessed in the session are limited to those associated with organization units in the organization context. For example, if the organization context is Stanford Hospital, a request to view encounters returns only those encounters at Stanford Hospital.

Tip: Two organization units should be placed into the same organization context when value objects associated with them are always accessed together. Otherwise, separate organization units should belong to different organization contexts.

Not all value objects are constrained by organization contexts. Table 4–13 lists value objects that are constrained by organization contexts through the corresponding organization attributes:

Table 4–13 Constrained Value Objects

Value Object	Organization Attribute
ActDefinition	OrganizationId
Appointment	OrganizationId
CareSite	OwningOrgId
ClinicalAct	MrnOrgPartyId
Encounter	ResponsibleHealthcareOrganizationIdentifier
EncounterGroup	ResponsibleHealthcareOrganizationIdentifier
Equipment	OwningOrgId
MedicalRecord	OrgId
Patient	Person.Domain.OrganizationId
PatientAppointment	Appointment.OrganiztionId
Person	Domain.OrganizationId
PrivilegeDefinition	OrgUnit.Id
StaffAssignment	OrgUnit.Id
StaffMember	ScopingOrgUnit.Id
StaffPosition	OrgUnit.Id
Verification	OrgUnit.Id
WaitList	WaitListOrganization.OrganizationId
WaitListEntry	WaitList.WaitListOrganization.OrganizationId

Note: Organization security is available for the Person value object only when the Flexible Person Domain Service is enabled (Section 4.2.4).

Prerequisites

- Implementing Person Services
- Implementing Organizations

Login

Log in to Oracle Applications as an administrative user.

Responsibility

System Administrator

Navigation

Table 4–14 summarizes the navigation paths used by this section:

Table 4–14 Navigation Paths: Creating and Associating Security Profiles

Function or Window	Navigation Path
Create Security Profile	`Navigator > Functions > Security > Profile`
Associate Security Profile with Responsibility	`Navigator > Functions > Profile > System`
Associate Security Profile with Responsibility Assignment	`Responsibility > Responsibility Search > View Responsibility Details > View Assigned Security Profiles > Assign Security Profiles`

4.1.10.1 Creating Security Profiles

Responsibility

HR Foundation

Steps

1. Navigate to the Navigator window.
2. Select the *Applications* > *Functions* link (under System Administrator).
3. Click the Functions tab in the new window.
4. Click *Security: Profile* from the right panel (Top Ten List) and select Open.
5. In the Security Profile window:
   - **Name:** Enter the name of the new security profile.
   - **Business Group:** Select the business group organization for the organization hierarchy.

See Also: Section 4.3, Implementing Organizations

- **Security Type:** Select *Security organizations by organization hierarchy and/or organization list.*
- **Organization Hierarchy:** Select the organization hierarchy.
- **Top Organization:** Define the sub-hierarchy to be included by selecting the top node.
- **Include Top Organization:** Check the flag to include the top node.
- **Organization Name column:** Fill in the table column with the following:
  - Organization outside the sub-hierarchy to be *included*.
  - Organizations within the sub-hierarchy to be ***excluded***.
- **Exclude Business Group:** Check the flag.

6. Repeat steps 1 through 4 as required.
7. Navigate to View > Request window:
   - Click *Submit a New Request.*
   - Select *Single Request.*
8. In the Submit Request window:
   - **Name:** Select Security List Maintenance.
   - **In the Parameters window:** Select All Security Profiles for *Generate lists for.*
   - Click Submit.

4.1.10.2 Associating Security Profiles with Responsibilities

Responsibility

System Administrator

Steps

1. Navigate to the Navigator window.
2. Select the *Applications* > *Functions* link (under System Administrator).
3. Click the Functions tab in the new window.
4. Click *Profile: System* from the right panel (Top Ten List) and select Open.
5. In the Find System Profile Values window:

- **Responsibility:** Select the responsibility.
- **Profile:** Select the system profile.
- Click Find button.
- In the System Profile Values window, select the security profile for the responsibility and system profile combination.

4.1.10.3 Associating Security Profiles with Responsibility Assignments

Responsibility

Healthcare Security Administrator

Steps

1. Navigate to the Responsibilities Search page and search for the responsibility.
2. Click the responsibility name to navigate to the View Responsibility Detail pages.
3. Select the account and navigate to the View Assigned Security Profile page.
4. Click the Add Security Profile button to navigate to the Assign Security Profiles page.
5. Click the torchlight to search for and select a security profile.
6. Click the Apply button.

4.2 Implementing Person Services

Person Services provide a central repository for person records, and facilitates system integration by providing a centralized person registry. It enables efficient identification of persons by providing a single view of a person—with multiple responsibilities across HTB and other external systems.

Prerequisites

Implementing Security Services:

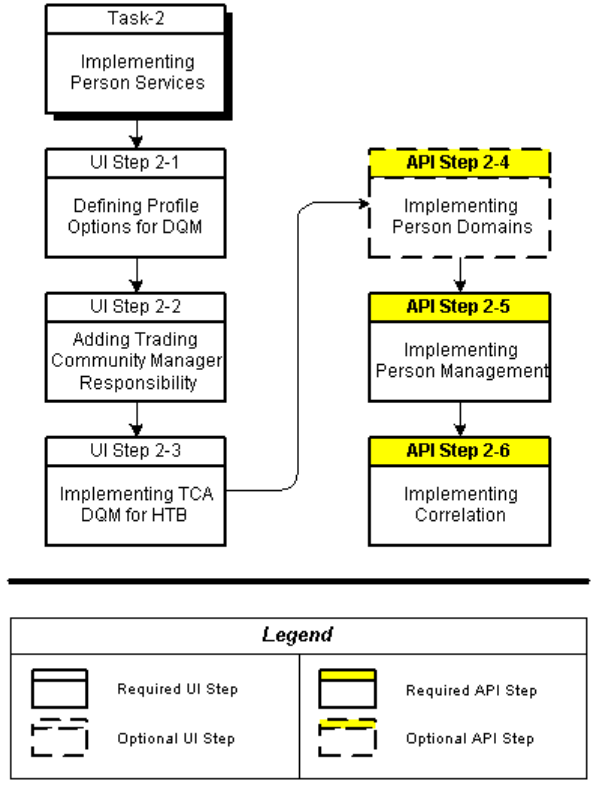
- An administrative security responsibility should be created and assigned all of the protected functions in Organization Services.
- An administrative account should be created, as a member of the security responsibility.

See Also: Section 4.1, Implementing Security Services

Procedures

Figure 4–10 provides an overview of the implementation process for Person Services:

Figure 4–10 Implementation Process: Person Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–3, HTB Implementation Procedures: Person Services

The following sections describe the implementation procedures for Person Services (referenced by Figure 4–10):

4.2.1 Defining Person Services Profile Options for DQM

Login

Log in to HTB as SYSADMIN, or use the administrative account created in Section 4.1.3.

Responsibility

Healthcare Configuration Administrator

Steps

This step precedes the actual TCA DQM setup. Use the Profile Options window user interface to set the Person Services profile option defaults listed by Table 4–15:

Table 4–15 Person Services Profile Options for DQM

Profile Option Name	Profile Option Code	Value
Address Status Custom Attribute	ADDRESS_STATUS_ ATTRIBUTE	CUSTOM_ATTRIBUTE2
Duplicate Threshold	DUPLICATE_THRESHOLD	85
DQM Identification	DQM_IDENTIFICATION_ NAME	Specify a value, such as California Driver's License
Identification Custom Attribute	IDENTIFICATION_ ATTRIBUTE	CUSTOM_ATTRIBUTE1
Phone Status Custom Attribute	PHONE_STATUS_ ATTRIBUTE	CUSTOM_ATTRIBUTE4
Phone Usage Type Attribute	PHONE_USAGE_TYPE_ ATTRIBUTE	CUSTOM_ATTRIBUTE3
Rule Name	RULE_NAME	Specify a value, such as HTB_DEDUP_RULE

Table 4–15 (Cont.) Person Services Profile Options for DQM

Profile Option Name	Profile Option Code	Value
Search Rule	SEARCH_RULE_NAME	Specify a value, such as HTB_SEARCH_RULE

See Also: Section 4.4, Implementing Profile Option Services

4.2.2 Adding Trading Community Manager Responsibility

Login

Log in to HTB as SYSADMIN, or use the administrative account created in Section 4.1.3.

Responsibility

System Administrator

Steps

Perform the following steps to add the Trading Community Manager Responsibility to the SYSADMIN user:

> **Note:** You can use either the SYSADMIN user name or the unique administrative account created in Section 4.1.3; the use of your unique user name is preferred.
>
> Throughout the following steps user SYSADMIN is used for ease of reference.

1. Log in to Oracle Applications as SYSADMIN and select System Administrator responsibility (automatic if only responsibility available); the System Administrator window is displayed by default.
2. Navigate to the Users window.
3. Select View > Query by Example > Enter or use the F11 function key; this lets you edit the form.
4. Enter SYSADMIN in the User Name text box and select View > Query by Example > Run (File Menu). This displays all available responsibilities for the SYSADMIN user.

5. If Trading Community Manager is not included in the list, do the following to add it as a new responsibility:
   - Select New (File Menu); this adds a new editable row in the Responsibilities section.
   - Click the three-dotted icon to the right of the highlighted first cell; this opens a Responsibility window.
   - Enter *Trading* next to the % sign in the Find field and click OK.
   - Select Trading Community Manager and click OK. This adds Trading Community Manager as a valid responsibility for `SYSADMIN`.
   - Select Save (File Menu) to save this change, and exit the application.

   Note: You can continue to the next step in the DQM implementation process as SYSADMIN, or select *File > Log on as Different User* to change your user name for the balance of the process.

4.2.3 Implementing TCA DQM for HTB

Person Services is employed to efficiently identify and correlate person records. It reengineers person identification data to meet the standards of content and formatting, and associates such data with relevant existing persons at the time of entry on a real-time basis. Such reengineering involves data cleansing, transformation, standardization and matching which is provided through Oracle Trading Community Architecture's (TCA) Data Quality Management tool (DQM).

Reference

Oracle Trading Community Architecture Data Quality Management User Guide

Responsibility

Trading Community Manager

Navigation

- Table 4–16 summarizes the navigation paths used by this section:

Table 4–16 Navigation Paths: Implementing TCA DQM for HTB

Function or Window	Navigation Path
Attributes and Transformation Functions Window	Data Quality Management > Setup > Attributes and Transformation Functions
Batch Duplicate Identification[1]	▪ Data Quality Management > Duplicate Identification > Batch Definition ▪ Data Quality Management > Duplicate Identification > Batch Review
Match Rules Window	Data Quality Management > Setup > Match Rules
Monitoring Requests	Data Quality Management > View Requests
Running DQM Staging Program (Submit New Request window)	Data Quality Management > Control > Requests > Run

[1] This function is used for unattended duplicate identification in batch mode. For further information, see: Oracle Trading Community Architecture Data Quality Management Guide, Batch Duplicate Identification section.

To implement TCA DQM for HTB, you must complete the steps described in the following sections:

- Section 4.2.3.1, Defining Attributes and Transformation Functions
- Section 4.2.3.2, Defining Match Rules

4.2.3.1 Defining Attributes and Transformation Functions

Perform the following steps to define attributes and transformation functions:

Login

Log in to Oracle Applications using the user name you defined in Section 4.1.3.

Responsibility

Trading Community Manager

Steps

1. Open the Define Attributes and Transformation Functions window.

2. Click the Party subtab (opened by default) and confirm that the following attributes are listed (Table 4–17):

Table 4–17 Party Attributes and Transformation Names

Attribute Name	Transformation Name	PL/SQL Function Name
Party Type	EXACT	HZ_TRANS_PKG.EXACT
Date of Birth	EXACT(DATE)	HZ_TRANS_PKG.EXACT_DATE
Gender	EXACT	HZ_TRANS_PKG.EXACT
Marital Status	EXACT	HZ_TRANS_PKG.EXACT
Person First Name	WR PERSON+CLEANSE EXACT SOUNDEX	HZ_TRANS_PKG.WRPERSON_CLEANSE HZ_TRANS_PKG.EXACT HZ_TRANS_PKG.SOUNDX
Person Last Name	WR PERSON+CLEANSE EXACT SOUNDEX	HZ_TRANS_PKG.WRPERSON_CLEANSE HZ_TRANS_PKG.EXACT HZ_TRANS_PKG.SOUNDX
Place of Birth	CLEANSE	HZ_TRANS_PKG.CLEANSE
Custom Attribute 1	EXACT	HZ_TRANS_PKG.EXACT
Custom Attribute 6	EXACT	HZ_TRANS_PKG.EXACT

3. For *each attribute* listed in Table 4–17, complete the following steps:

   - If the attribute is missing, select `File > New` (while in the Attribute Name region). This creates a new row with an LOV icon in the right side of the window. Click the LOV icon, and select the missing attribute. The attribute appears under Attribute Name.
   - For Custom Attribute 1:

Under Field Name	Enter
User Defined Name	Identification
Custom Procedure	CTB_TRANSFORMATION_FUNCTIONS.get_person_identification

   - For Custom Attribute 6:

Under Field Name	Enter
User Defined Name	DOMAIN ID

Under Field Name	Enter
Custom Procedure	CTB_TRANSFORMATION_FUNCTIONS.get_domain_id

- If any of the transformations is missing for the corresponding attribute in the Transformation Name region of the window, select File > New; a new transformation row appears. Enter the Transformation Name and the corresponding PL/SQL Function Name for each missing transformation.
- Confirm that both the Active and Acquisition check boxes are marked for each of these transformations (for each attribute).
- Save your work.

4. Click the Address subtab and confirm that the following transformations are listed (Table 4–18):

Table 4–18 Address Attributes and Transformations

Attribute Name	Transformation Name	PL/SQL Function Name
Address 1	CLEANSED EXACT	HZ_TRANS_PKG.CLEANSE HZ_TRANS_PKG.EXACT
Address 2	CLEANSE	HZ_TRANS_PKG.CLEANSE
Address 3	CLEANSE	HZ_TRANS_PKG.CLEANSE
Address 4	CLEANSE	HZ_TRANS_PKG.CLEANSE
City	CLEANSE	HZ_TRANS_PKG.CLEANSE
Country	EXACT	HZ_TRANS_PKG.EXACT
County	CLEANSE	HZ_TRANS_PKG.CLEANSE
Postal Code	EXACT	HZ_TRANS_PKG.EXACT
Province	CLEANSE	HZ_TRANS_PKG.CLEANSE
State	WR STATE + CLEANSE	HZ_TRANS_PKG.WRSTATE_CLEANSE
Custom Attribute 2	EXACT	HZ_TRANS_PKG.EXACT

5. For *each attribute* listed in Table 4–18, complete the following steps:
   - If the attribute is missing, select File > New (while in the Transformations region). This creates a new row with an LOV icon in the right side of the window. Click the LOV icon, and select the missing attribute. The attribute now appears under Attribute Name.
   - For Custom Attribute 2:

Under Field Name	Enter
User Defined Name	`Address Status`
Custom Procedure	`CTB_TRANSFORMATION_FUNCTIONS.get_address_status`

- If any of the transformations are missing for the corresponding attribute in the Transformation Name region of the window, select `File > New`; a new transformation row appears. Enter the Transformation Name and the corresponding PL/SQL Function Name for each missing transformation.
- Confirm that both the Active and the Acquisition check boxes are marked, for each of these transformations.
- Save your work.

6. Click the Contact Point subtab and confirm that the following attributes are listed (Table 4–19):

Table 4–19 Contact Point Attributes and Transformations

Attribute Name	Transformation Name	PL/SQL Function Name
Phone Area Code	`EXACT`	`HZ_TRANS_PKG.RM_SPLCHAR`
Phone Country Code	`EXACT`	`HZ_TRANS_PKG.RM_SPLCHAR`
Phone Extension	`EXACT`	`HZ_TRANS_PKG.RM_SPLCHAR`
Phone Number	`EXACT`	`HZ_TRANS_PKG.RM_SPLCHAR`
Phone Line Type	`EXACT`	`HZ_TRANS_PKG.EXACT`
Custom Attribute 3	`EXACT`	`HZ_TRANS_PKG.EXACT`
Custom Attribute 4	`EXACT`	`HZ_TRANS_PKG.EXACT`

7. For each attribute listed in Table 4–19, complete the following steps:
   - If the attribute is missing, select `File > New` (while in the Transformations region of the window). This creates a new row with an LOV icon in the right side of the window. Click the LOV icon, and select the missing attribute. The attribute appears under Attribute Name.
   - For Custom Attribute 3:

Under Field Name	Enter
User Defined Name	`Phone Usage Type`
Custom Procedure	`CTB_TRANSFORMATION_FUNCTIONS.get_phone_usage_type`

- For Custom Attribute 4:

Under Field Name	Enter
User Defined Name	`Phone Status`
Custom Procedure	`CTB_TRANSFORMATION_FUNCTIONS.get_phone_status`

- If any transformation is missing for its corresponding attribute in the Transformations region of the window, select `File > New.` A new transformation row appears. Enter the Transformation Name and the corresponding PL/SQL Function Name for each missing transformation.
- Confirm that both the Active and Acquisition check boxes are marked for each of these transformations.
- Save your work.

8. Run the DQM Staging Program.

- Navigate to the Trading Community Manager window and click the Functions tab.
- Navigate to `Control > Request > Run.`
- Select Single `Request > OK`.
- From the LOV on the right side of the Request Name, select DQM Staging Program and enter the following Parameters:

Parameter	Enter
Number of Parallel Staging Workers	1
Staging Command:	Select STAGE_ALL_DATA
Continue Previous Execution?	Click No.

- Click OK for Parameters, and Submit for Request (note the number for your request).

9. To monitor the status of your request, you can select `View > Requests`; the Find Requests window appears. Select All My Requests to view and monitor your specific request as well as children requests it generates. From there, you can view log files if your requests fail to complete successfully. When the DQM Staging Program finishes running, you can define Match Rules.

4.2.3.2 Defining Match Rules

Two separate Match Rules must be set:

- Deduplication Rule—used during creation and correlation of persons in HTB
- Search Rule—used for real-time searches involving Person objects

4.2.3.2.1 Setting Deduplication Rule

Steps

1. Navigate to the Match Rules window.
2. Assign a name to your Rule: *HTB_DEDUP_RULE,* and provide an optional description. *The name must match the corresponding name defined in the Profile Options (*Table 4–15*).*
3. Change Purpose from the default value *Search* to *Identify Duplicates*.
4. Select Attribute Match option: *Match Any Attribute*
5. Click the Acquisition subtab and complete the following steps *for each attribute* listed in Table 4–20:
   - Click the Attribute Name row; LOV dots appear to the right.
   - Click the dots and select an attribute.

     Note: Some Attribute Names may appear more than once for different Entities. Make sure that you select attribute names for the correct Entity.

   - For each attribute, enter the transformations from Table 4–20 into the Transformations region; LOV dots appear to the right of the Transformation Name.
   - Click the Filter checkbox for the attributes Party Type and Domain ID.
   - Save your work.

Table 4–20 Acquisition Attributes and Transformations: Deduplication Rule

Attribute Name	Entity	Transformation Name
Phone Number	CONTACT_POINTS	▪ EXACT
Party Type	PARTY	▪ EXACT

Table 4–20 (Cont.) Acquisition Attributes and Transformations: Deduplication Rule

Attribute Name	Entity	Transformation Name
Domain ID	PARTY	▪ EXACT
Date of Birth	PARTY	▪ EXACT(DATE)
Person Last Name	PARTY	▪ WR PERSON+CLEANSE ▪ SOUNDEX
Address 1	PARTY_SITES	▪ CLEANSED ▪ EXACT
Postal Code	PARTY_SITES	▪ EXACT

6. Click the Scoring subtab and complete the following steps *for each attribute* listed in Table 4–21:
   - Click an Attribute Name row; LOV dots appear to the right; click the dots and select an attribute; enter the corresponding score.
   - Enter the transformations from Table 4–21 in the Transformations region. LOV dots appear to the right of the Transformation Name.
   - Save your work.

Table 4–21 Scoring Attributes and Transformations: Deduplication Rule

Attribute Name	Entity	Transformation Name	Score	Weight %	Similarity (Exact)	%
Phone Area Code	CONTACT_POINTS	EXACT	5	100	Checked	NA
Phone Number	CONTACT_POINTS	EXACT	10	100	Checked	NA
Date of Birth	PARTY	EXACT(DATE)	15	100	Checked	NA
Gender	PARTY	EXACT	5	100	Checked	NA
Person First Name	PARTY	WR PERSON+CLEANSE	5	90	Unchecked	90
Person First Name	PARTY	EXACT	5	100	Checked	NA
Person First Name	PARTY	SOUNDEX	5	70	Checked	NA
Person Last Name	PARTY	WR PERSON+CLEANSE	20	90	Unchecked	90
Person Last Name	PARTY	EXACT	20	100	Checked	NA

Table 4–21 (Cont.) Scoring Attributes and Transformations: Deduplication Rule

Attribute Name	Entity	Transformation Name	Score	Weight %	Similarity (Exact)	%
Person Last Name	PARTY	SOUNDEX	20	70	Checked	NA
Place of Birth	PARTY	CLEANSE	5	90	Unchecked	90
Address 1	PARTY_SITES	CLEANSE	15	90	Unchecked	90
Address 1	PARTY_SITES	EXACT	15	100	Checked	NA
City	PARTY_SITES	CLEANSE	5	90	Unchecked	90
Postal Code	PARTY_SITES	EXACT	5	100	Checked	NA
Province	PARTY_SITES	CLEANSE	5	90	Unchecked	90
State	PARTY_SITES	WR STATE+CLEANSE	5	100	Checked	NA

7. Enter Match Threshold = 77.

> **Note:** The Match Threshold value defines the system minimum threshold. DQM returns all records that are matched above this value. The records that score above the DUPLICATE_THRESHOLD value are considered duplicates.

8. Save your work.
9. Click Compile. A message indicates that your Match Rule has been compiled successfully.

4.2.3.2.2 Setting Search Rule

Steps

1. Navigate to the Match Rules window.
2. Assign a name to your rule: *HTB_SEARCH_RULE*, and provide an optional description. *The name must match the corresponding name defined in the Profile Options (*Table 4–15*).*
3. Leave Purpose at default value *Search*.
4. Select Attribute Match option *Match Any Attribute*.

5. Click the Acquisition subtab and complete the following steps *for each attribute* listed in Table 4–22.

   - Click the Attribute Name row. LOV appears to the right.
   - Click the dots and select an attribute.

   Note: Some Attribute Names may appear more than once for different entities. Make sure that you select attribute names for correct entity.

   - For each attribute, enter the transformations from Table 4–22 into the transformations region; LOV dots appears to the right of the transformation name.
   - Click the Filter checkbox for the attributes Party Type and Domain ID.
   - Save your work.

Table 4–22 Acquisition Attributes and Transformations: Search Rule

Attribute Name	Entity	Transformation Name
Phone Number	CONTACT_POINTS	▪ EXACT
Party Type	PARTY	▪ EXACT
Domain ID	PARTY	▪ EXACT
Date of Birth	PARTY	▪ EXACT(DATE)
Identification	PARTY	▪ EXACT
Person Last Name	PARTY	▪ WR PERSON+CLEANSE ▪ SOUNDEX
Address 1	PARTY_SITES	▪ CLEANSE

6. Click the Scoring subtab and complete the following steps for each attribute listed in Table 4–23.

   - Click an Attribute Name row. LOV dots appear to the right; click the dots and select an attribute; enter the corresponding score.
   - Enter the transformations from Table 4–23 in the Transformations region. LOV dots appear to the right of the Transformation Name.
   - Save your work.

Table 4–23 Scoring Attributes and Transformations: Search Rule

Attribute Name	Entity	Transformation Name	Score	Weight %	Similarity (Exact)	%
Phone Area Code	CONTACT_POINTS	EXACT	3	100	Checked	NA
Phone Country Code	CONTACT_POINTS	EXACT	3	100	Checked	NA
Phone Extension	CONTACT_POINTS	EXACT	3	100	Checked	NA
Phone Number	CONTACT_POINTS	EXACT	3	100	Checked	NA
Phone Status	CONTACT_POINTS	EXACT	3	100	Checked	NA
Phone Usage Type	CONTACT_POINTS	EXACT	3	100	Checked	NA
Phone Line Type	CONTACT_POINTS	EXACT	3	100	Checked	NA
Date of Birth	PARTY	EXACT (DATE)	5	100	Checked	NA
Gender	PARTY	EXACT	3	100	Checked	NA
Identification	PARTY	EXACT	10	100	Checked	NA
Marital Status	PARTY	EXACT	3	100	Checked	NA
Person First Name	PARTY	WR PERSON+CLEANSE	5	90	Unchecked	90
Person First Name	PARTY	SOUNDEX	5	70	Checked	NA
Person Last Name	PARTY	WR PERSON+CLEANSE	10	90	Unchecked	90
Person Last Name	PARTY	SOUNDEX	10	70	Checked	NA
Place of Birth	PARTY	CLEANSE	3	90	Unchecked	90
Status	PARTY	EXACT	3	100	Checked	NA
Address 1	PARTY_SITES	CLEANSE	5	90	Unchecked	90
Address 2	PARTY_SITES	CLEANSE	3	90	Unchecked	90
Address 3	PARTY_SITES	CLEANSE	3	90	Unchecked	90
Address 4	PARTY_SITES	CLEANSE	3	90	Unchecked	90
Address Status	PARTY_SITES	EXACT	3	100	Checked	NA
City	PARTY_SITES	CLEANSE	7	99	Unchecked	90
Country	PARTY_SITES	EXACT	3	100	Checked	NA
County	PARTY_SITES	CLEANSE	3	90	Unchecked	90

Table 4–23 (Cont.) Scoring Attributes and Transformations: Search Rule

Attribute Name	Entity	Transformation Name	Score	Weight %	Similarity (Exact)	%
Postal Code	PARTY_SITES	EXACT	3	100	Checked	NA
Province	PARTY_SITES	CLEANSE	3	90	Unchecked	90
State	PARTY_SITES	WR STATE+CLEANSE	3	90	Unchecked	90

7. Enter match threshold = 70.

> **Note:** The Match Threshold value defines the system minimum threshold. DQM returns all records that are matched above this value.

8. Save your work.
9. Click Compile; a message indicates that your Match Rule has been compiled successfully.

4.2.4 Implementing Person Domain Service

Person Domain Service is an extension of Person Services that provides support for optional partitioning of person demographic data within an installation. Such partitioning can be scoped by the Healthcare organization that owns the data, and additionally at the more refined level based on the person role associated with demographics data (e.g., patient, staff members, and so on).

Implementing Person Domain Service is optional for most users, and is required only within an environment that contains multiple independent internal enterprises within the same installation (e.g., "community model"). Using Person Domains can be chosen when there are legal or business needs for partitioning person demographics – for example, HR privacy guidelines that impose strict set of restrictions on sharing sensitive employee demographics information. However, when only one set of person demographics per HTB person is required, then Domain Service does not need to be implemented.

Domain Service provides complete domain configuration and edit functions, including create, update and delete. Once the user implements Person Domain service, two additional functions become available within Person Management

service: for associating existing person records with a domain, and for clearing any domain association, respectively.

In addition, HTB Person Services provide migration scripts for migrating existing customer data to domain-configured setup. This procedure can be also used for bulk-association of records that have been created outside of HTB or batch loaded.

Reference

Oracle Javadoc for HTB

Table 4–24 lists the flexible person domain service methods:

Table 4–24 Service and Methods: Implementing Person Domain Service

Level	Detail
Package	`oracle.apps.ctb.admin.person`
Class	`DomainService`
Methods	▪ `createDomain` ▪ `deleteDomain` ▪ `findDomains` ▪ `getDomain` ▪ `updateDomain`

Prerequisites

- Implementing Organizations
- Setting Profile Option: The following Person Domain Services Profile Option must be set:
  `CTB: Enabled Domain`.

> **Note:** Setting Person Domain Services Profile Option:
>
> - Name: `CTB: Enabled Domain`
> - Profile Option Code: `CTB_PI_ENABLED_DOMAIN`
> - Value: `Y`

See Also:

- Section 4.3, Implementing Organizations
- Section 4.4, Implementing Profile Option Services

Login

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

4.2.4.1 Creating Person Domains

Use the `createDomain` method to create a Person Domain and assign a system generated Domain ID. A domain definition must include a valid organization ID and at least one Domain Role. Valid domain organizations are organization units that have an organization role of Enterprise associated with them. Valid domain roles are seeded in the `CTB_DOMAIN_ROLE` concept list, and include the following:

- Patient
- Healthcare Professional
- Guarantor
- Contact

Note: The system does not create overlapping domains (domains that include the same combination of an organization ID and a role.)

4.2.4.2 Deleting Person Domains

Use the `deleteDomain` method to delete an existing Person Domain. A domain can only be deleted if no associated person records exist within the domain. Therefore, before you can delete a domain, you must first clear the association of any existing records with that domain.

4.2.4.3 Updating Person Domains

Use the `updateDomain` method to update an existing Person Domain. This method supports the updating of the description and the addition of domain roles. The updated domain cannot contain the same role as another domain if both are scoped by the same organization.

> **Note:** You can add valid roles, but you can not delete roles from a domain definition.

4.2.4.4 Searching for Person Domains

- Use the `getDomain` method to retrieve an existing domain by the domain ID.
- Use the `findDomains` method to retrieve existing domains that match the given search criteria, which may include the description (full or partial), Organization ID, or Domain Role Code.

4.2.5 Implementing Person Management

Person Management provides complete person edit functions including create, update, inactivate, and delete. To facilitate access to all person attributes, Person Management maintains a consistent and duplicate-free person repository. The repository contains demographic attributes and maintains internal links to the specific data necessary for management of all records associated with a person, including user, staff, patient, and encounter records. This enables centralized management of personal data to ensure duplicates resolution, and eliminates data redundancy while providing secure and efficient storage of sensitive information—such as employment and medical records.

Person Management also supports management of incomplete person records. It may be necessary to create incomplete person records in situations where a person record must exist, but insufficient data is available to fully qualify that person as a unique individual. Examples of situations where this may occur are in the emergency room or in communications where there are language barriers. In such cases, the system does not check whether the user provided the required information to create a person record.

Reference

Oracle Javadoc for HTB

Table 4–25 lists the principal person service and methods:

Table 4–25 Service and Methods: Implementing Person Management

Level	Detail
Package	`oracle.apps.ctb.admin.person`
Class	`ManagePersonService`
Methods	▪ `associatePersonDomain` ▪ `clearPersonDomain` ▪ `createPerson` ▪ `mergePerson` ▪ `updatePerson`

Prerequisites

- Setting Profile Option: Person Services Profile Options must be set
- Implementing TCA DQM for HTB
- Implementing Person Domain Service—for the following methods:
  - *associatePersonDomain*
  - *clearPersonDomain*

See Also:

- Section 4.2.1, Defining Person Services Profile Options for DQM
- Section 4.2.3, Implementing TCA DQM for HTB
- Section 4.2.4, Implementing Person Domain Service

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

4.2.5.1 Associating Person Domains

Use the `associatePersonDomain` method to associate existing person records with a person domain. A person can be associated with a domain only if both of the following conditions are met:

- The person is not already associated with a domain.
- The person does not have any dependent data outside of the scope of the domain.

If a person must be associated with more than one domain, or if the record has conflicting dependent data, the user must resolve this manually by creating a copy of the person record and separating the dependent data before using the *associatePersonDomain* method.

> **Note:** HTB Person Services provides scripts for migrating existing customer data to domain-configured setup.

4.2.5.2 Clearing Person Domains

Use the `clearPersonDomain` method to clear an existing association of a person object with a person domain. This may be necessary when a domain association has been made in error, or where an organizational restructure occurs that impacts the current definitions—dictating the subsequent need to reassociate persons to different domains.

4.2.5.3 Creating Persons

Use the `createPerson` method to create new person records. A person record is the foundation for creating a staff member, patient, or user. Basic information such as legal name and address and demographic information such as gender, date of birth, birthplace, and identifier are specified in the creation of a person record. This method initially checks if sufficient minimum demographic data is specified. This is a requirement for duplicate identification in the person repository. If this requirement is not met, the newly created person record is marked as incomplete.

The `createPerson` method ensures that the new person record is not a duplicate of an existing person record, unless you explicitly request no duplicate matching by setting the Run Match flag to *No*. In this case matching is skipped, and the newly created person record is marked as not matched. Otherwise, the matching process searches for potential duplicates in the repository and the system restricts the

creation of a new record if a duplicate person record does exist. To override this restriction, you can set the Allow Duplicates flag to *Yes*.

4.2.5.4 Updating Persons

Use the `updatePerson` method to update existing person records. The business rules governing the creation of persons also apply to the updating of persons.

4.2.5.5 Merging Persons

Use the `mergePerson` method to merge two person records. The merge request is submitted to TCA through a concurrent program, in either synchronous or asynchronous mode. When the mode is synchronous, control from the concurrent program is returned only when the merge execution is finished. If the mode is asynchronous, the concurrent process continues in the background after the merge request ID is returned. This ID can be used to check the status of the concurrent program using the Oracle Applications Concurrent Request View window user interface.

Specify the surviving (master) person ID, the donor person ID and the desired merge mode.

Note: If Person Domain Service is implemented, persons associated with different domains cannot be merged together.

See Also:

- Section 4.1, Implementing Security Services
- Section 4.2.4, Implementing Person Domain Service
- Section 4.8.4, Implementing Staff Management
- Section 4.8.9, Implementing Patient Registration
- Appendix J, Party Merge Procedures
- *Oracle Trading Community Architecture Party Merge User Guide*

4.2.6 Implementing Correlation

HTB Correlation maintains a cross-index of person identifiers across all source systems. Through cross-referencing capabilities, person records in disparate systems are correlated and synchronized. The use of correlated person data can range from

completely centralized management of person attributes in a master index serving multiple feeder systems to loosely coupled persons cross-referenced between source systems.

Person repositories batch loaded from other systems can also be correlated, generating internal identifiers and cross-referencing external identifiers as needed.

Also supported is the internal correlation of person records that links related record across all person domains within the installation. This includes matching person records against the master view that represents the source of person demographics based on the user-defined order of domain precedence.

Reference

- *Oracle Javadoc for HTB*
- *TCA User Guide, Customer Import*
- *Oracle TCA Data Quality Management User Guide Release 11i*

Table 4–26 lists the principal correlation service methods:

Table 4–26 Service and Methods: Implementing Correlation

Level	Detail
Package	`oracle.apps.ctb.admin.person`
Class	`CorrelatePersonService`
Methods	▪ `batchCorrelate` ▪ `correlateExternalPerson` ▪ `createCorrelateData` ▪ `crossReferenceCheck` ▪ `deleteExternalPerson` ▪ `externalMergeCrossRefPerson` ▪ `findCrossRefByPersonID` ▪ `findPersonByExternalID` ▪ `transferCorrelationData`

Prerequisites

- Implementing TCA DQM for HTB
- Setting Profile Option: Person Services profile options must be set

See Also:

- Section 4.2.1, Defining Person Services Profile Options for DQM
- Section 4.2.3, Implementing TCA DQM for HTB

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

4.2.6.1 Correlating External Persons

Use the `correlateExternalPerson` method to correlate an external person record with a record in HTB. If a matching record does not exist in HTB, a new record is created and correlated with the external system record.

Specify the person attributes, the external system, the external system person identifier, and the distinct threshold.

See Also: Section 4.2.5, Implementing Person Management, for more information about the matching process.

4.2.6.2 Creating Correlation Data

Use the `createCorrelateData` method to correlate an existing person record (registered in Person Services) with an external person record. Note that no matching process is invoked. Specify an external system, an external system identifier for the person, and the Person Services identifier for the person. An entry is created in the HTB cross-reference registry to represent this correlation.

4.2.6.3 Correlating Persons in Batch

Use the `batchCorrelate` method to correlate a batch or set of person records. When external person records are loaded into the system, they are compared with

existing Person Services records to identify duplicates. The records are subject to the match and *deduplication* process, as for the normal creation of person records. Any non-duplicate records are created as person records in Person Services, and they are correlated with the external person identifiers via the creation of entries in the cross-reference registry.

Processing the batch data includes two components: DQM, and the `batchCorrelate` method. In addition, you can optionally use the `crossReferenceCheck` method to perform a cross-reference check on the external data—after entering the batch data, but before calling `CorrelateBatch`, to filter out any records that have previously been correlated.

Both the uncorrelated person data that has been imported into TCA and the current active person data reside in the TCA repository. However, the uncorrelated person data should be excluded from processing by other applications until it has been processed through DQM and Person Services APIs. Accordingly, the batch data should be marked for exclusion by using the combination of the application ID and the TCA classification for each uncorrelated record. A null TCA classification in combination with the CTB application ID labels a record as *unavailable*. After the record has been correlated, the TCA classification is updated to *available*.

Upon completion of the classification and data sharing steps, the DQM batch identification process can be defined and processed. After processing the batch data through DQM, the client calls `CorrelateBatch`, passing in the batch ID and the optional distinct threshold as parameters.

The following profile options must be defined for batch correlation:

- `EXTERNAL_SYSTEM_ID_ATTRIBUTE`

  Defines which attribute column is used to store external person system ID in `HZ_PARTIES (Attribute1` column, for example).

- `EXTERNAL_PERSON_ID_ATTRIBUTE`

  Identifies which attribute column is used to store external person ID in `HZ_PARTIES (Attribute2` column, for example).

Note: Match rules required to be defined for batch matching may vary from those defined for real-time matching in Section 4.2.3; see the *Oracle TCA Data Quality Management User Guide Release 11i* (Batch Duplicate Identification section) for guidance in formulating match rules for batch correlation.

See Also:

- Section 4.2.5, Implementing Person Management
- Section 4.4.1, Implementing Profile Options
- *Oracle TCA Data Quality Management User Guide Release 11i*

4.2.6.4 Finding Cross-Reference Data for Person Record

Use the `findCrossRefByPersonId` to find one or more cross-reference records in the external cross-reference registry associated with the given internal person identifier.

4.2.6.5 Finding Person Record by External ID

Use the `findPersonByExternalID` method to find a person record in HTB Person Registry given external person and external system identifiers. Given the external identifiers, the cross-reference registry is queried to retrieve the Person Services person identifier, which in turn is used to return the Person Services person record associated with that identifier.

4.2.6.6 Checking Cross-Reference Registry for Batch Records

Use the `crossReferenceCheck` method to check a batch of external person records against the cross-reference registry, to determine if any of the records have been previously correlated. The status of any records from the batch with entries in the cross-reference registry is updated to *Previously Correlated*. This optimizes the correlation process, so that the previously-correlated records do not need to be resubmitted for correlation.

4.2.6.7 Responding to External Merge of Person Records

Use the `externalMergeCrossRefPerson` method to update and verify the cross-reference registry as a result of a merge between two persons in an external system. Specify the identifiers of the external merge candidates, the external system, and the identifier of the resulting external merged record. The external merge candidate identifiers are used to check the cross-reference registry and locate their associated Person Services person identifiers. In the case where both external merge candidates are found to be correlated with two different internal Person Services records, an error is raised (due to potential internal duplicates). You can further resolve this by using other `CorrelatePersonService` methods:

- `createCorrelateData`

- `transferCorrelationData`
- `deleteExternalPerson`

4.2.6.8 Transferring Correlation Data

Use the `transferCorrelationData` method to transfer the correlation for a Person Services person from one external person identifier to another external person identifier. Specify the external system, the current external system identifier, and the new external system identifier to which to transfer the correlation.

4.2.6.9 Deleting Cross-Reference Entry

Use the `deleteExternalPerson` method to delete an entry in the cross-reference registry. This may be necessary as a response to a merge or deletion of an external person record. Specify the external system and the external system identifier to locate the associated Person Services identifier.

4.3 Implementing Organizations

Organization Management lets you create and maintain an infrastructure to model healthcare organization structures and the relationships between organization units within such structures. This HTB component lets users define and classify organization units, and associate them with hierarchies.

Reference

- *Oracle Javadoc for HTB*
- *Using Oracle HRMS - The Fundamentals (US)*
- *Configuring, Reporting and System Administration in Oracle HRMS*

Prerequisites

- Implementing Security Services:
  - An administrative security responsibility must be created and assigned all of the protected functions in Organization Services.
  - An administrative account must be created, as a member of the security responsibility.
- Implementing Profile Option Services: The HR: Business Group and the HTB Business Group profile options must be set to the same business group organization.

See Also:

- Section 4.1, Implementing Security Services
- Section 4.4, Implementing Profile Option Services

User Interface Support

Although this is principally an API-based implementation procedure, you can also use the Oracle Human Resources (HRMS) **Graphic User Interface** to set up and maintain an HTB-recognized organization, hierarchy, and location—defining most of the required attributes (except for those stored in HTB tables); you can define the balance of the attributes using HTB APIs. The applicable HRMS windows include the following:

- Define Organization
- Additional Information

- Hierarchy Definition
- Define Location

Use the Define Organization window to find an existing organization or create a new one. From this window you can navigate to the Additional Information window, which lets you define additional attributes based on the organization classification (equivalent to the HTB Organization Role). Each organization classification can have one or more additional organization information types associated with it. In addition, a particular organization information type can be shared by multiple organization classifications, so that the data entered for a shared type appears for multiple organization classifications.

Use the Organization Hierarchy window to define the basic hierarchy attributes and the parent/child relationships that make up the hierarchy elements. Although HRMS hierarchy elements can be created or updated in the current version, HTB APIs enforce that population of the hierarchy elements; subsequent changes to these elements are made through the use of `performAction` functions— creating a new version and submitting an array of additions, moves, and removals to and from the hierarchy. An HTB user must create a new version and then populate the hierarchy. Note that the Hierarchy Type must be defined through HTB APIs *after* the hierarchy has been created.

Use the Location and Location Extra Information windows to define organization internal addresses.

Note: You must associate an HTB hierarchy type with an HR organization hierarchy using an HTB API. Some HTB organization data, such as contact points, short name, description, status, hours of operation, identifications, and external organization identifier (called *org id*) are not available through the HRMS user interface. To access certain HTB attributes, you must assign a Healthcare Classification (HTB organization role—called *org role*) to an HR organization; this is done automatically when you use HTB APIs.

The following HTB business validations are also enforced through the HRMS user interface:

- You cannot delete organizations, hierarchies, or locations with a Healthcare classification.[1]

- You must remove organization units from all hierarchies before terminating them.
- The following organization roles (HR organization classifications) are mutually exclusive for any single organization unit:
  - Enterprise
  - Facility
  - Practice Setting
  - Business Unit

Login

- To perform this procedure as an API-based implementation procedure, log in to HTB using the administrative account created in the Prerequisites section.
- To perform this procedure using the HRMS graphic user interface, log in to Oracle Applications as an HR Administrator.

See Also:

- Appendix A, HTB Session Service
- User Interface Support

Responsibility

HR Administrator

Navigation

To perform this procedure as an API-based implementation procedure,

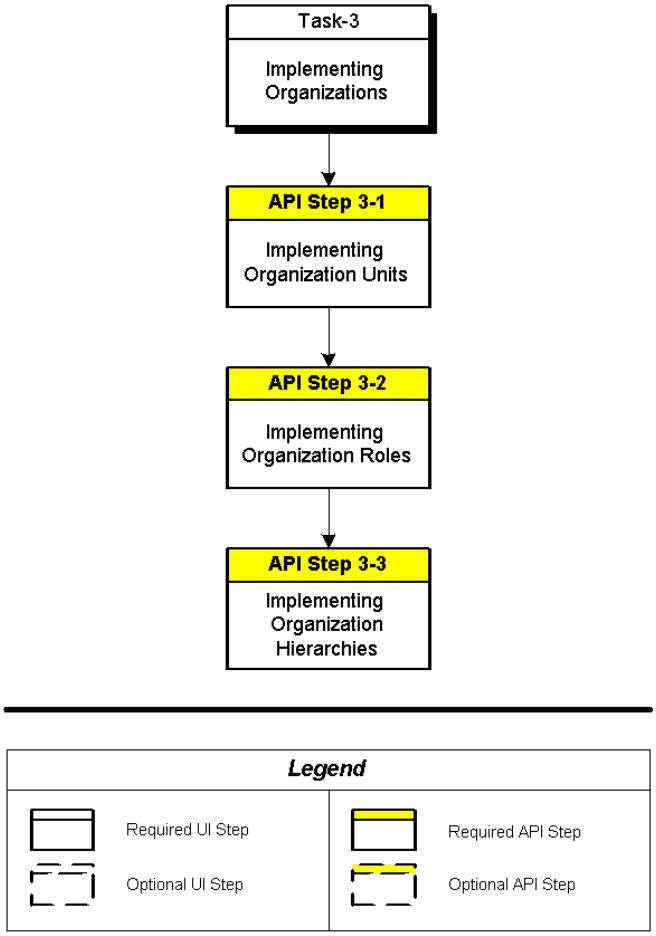
See: Appendix A, HTB Session Service

Procedures

Figure 4–11 provides an overview of the implementation process for Organizations:

[1] This is a physical delete in HRMS.

Figure 4–11 Implementation Process: Organizations



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–4, HTB Implementation Procedures: Organizations

The following sections describe the implementation procedures for Organizations (referenced by Figure 4–11):

- Section 4.3.1, Implementing Organization Units
- Section 4.3.2, Implementing Organization Roles
- Section 4.3.3, Implementing Organization Hierarchies

4.3.1 Implementing Organization Units

Reference

Oracle Javadoc for HTB

Table 4–27 summarizes Organization Unit service and methods:

Table 4–27 Service and Methods: Organization Units

Level	Detail
Package	`oracle.apps.ctb.admin.org`
Class	`OrgUnitService`
Methods	▪ `createOrgUnit` ▪ `findOrgUnitsBySearchCriteria` ▪ `getOrgUnitByExternalId` ▪ `getOrgUnitById` ▪ `terminateOrgUnit` ▪ `updateOrgUnit`

4.3.1.1 Creating Organization Units

Use the `createOrgUnit` method to create a new organization unit. Organization units always exist as stand-alone objects, and are typically associated with one or more organization hierarchies. Creating an organization unit involves specifying attributes such as name, existence time, and location.

- **Organization Unit Names**

  Organization unit names must be unique within a business group. If the organization itself is a business group, its name must be unique across all business groups. An organization unit long name is compared with other

organization unit long names for uniqueness, as are organization unit short names.

Note: A business group is an organization in Oracle HRMS that can be structured to serve multiple purposes, such as a legislative partition. Business groups hold organization units, hierarchies, and addresses. Within HTB, a single business group serves as the default for the creation of organization units, hierarchies, and addresses, and is specified by the profile options HR: Business Group, and HTB Business Group. For more information about creating a business group, see *Using Oracle HRMS - The Fundamentals (US).*

- Organization Unit Existence times

  An organization unit can be associated with an organization hierarchy or referenced by other HTB business entities within two date/time stamps—its *existence time low* and *existence time high* (equivalent to the start and end dates and times of the life of an organization).

  HTB enforces several rules on date validation. An organization unit's existence time low must be earlier than its existence time high. This is generally applicable to any business entity with dates. When participating in an enterprise type hierarchy, a child organization unit's existence time low must be greater than or equal to its parent's existence time low, and its existence time high must be less than or equal to its parent's existence time high. Organization subcomponents that have start and end date attributes also follow this convention.

- **Organization Unit Subcomponents**
  - **Locations and Location Uses**

    An organization can have one location and up to four location uses. The location represents the physical address of the organization, while the location use indicates the context of that address, such as a bill-to address. One named location can be shared by multiple organizations, but an organization cannot reference multiple locations.

  - **Contact Points**

    An organization can have zero to many contact points, which can be a phone number, e-mail address, or Web address.

- **Identifications**

  An organization unit can have zero to many identifications, issued by external organizations. Each identification consists of a type, number, issuing location, issuing organization, and validity times.

- **Hours of Operation**

  Indicates the days and times during which the organization unit operates.

4.3.1.2 Updating Organization Units

Use the `updateOrgUnit` method to update existing organization units. The same business rules associated with name uniqueness, existence times, and subcomponents governing the creation of organization units also apply during the update of organization units.

4.3.1.3 Terminating Organization Units

Use the `terminateOrgUnit` method to terminate an organization after it has been removed from all hierarchies, by specifying its existence time high, which can be in the past, present, or future. Once an organization unit is terminated, it should not be used in HTB transactions.

4.3.1.4 Searching for Organization Units

Use the `findOrgUnitsBySearchCriteria` method to find existing organization units by searching `SearchCriteria` attributes.

Use the `getOrgUnitById` method to retrieve a single organization unit by Organization Unit ID.

4.3.2 Implementing Organization Roles

Reference

Oracle Javadoc for HTB

Table 4–28 summarizes Organization Role service and methods:

Table 4–28 Service and Methods: Organization Roles

Level	Detail
Package	`oracle.apps.ctb.admin.org`

Table 4–28 (Cont.) Service and Methods: Organization Roles

Level	Detail
Class	`OrgUnitService`
Methods	▪ `createOrgRoleAssociation` ▪ `findOrgRoleAssociationsBySearchCriteria` ▪ `updateOrgRoleAssociation`

4.3.2.1 Organization Roles

Each organization unit can have multiple roles, which define its place in the enterprise hierarchy, its function within the enterprise, and its relationship with other organization units. HTB supports the following organization roles, which are mutually exclusive:

- An **enterprise** is the principal business entity that defines a healthcare organization. It maintains a **patient** registry, a staff registry, a user registry, and other HTB entities. This organization role must be assigned to the highest organization unit in the enterprise hierarchy. The HTB enterprise is analogous to a business group within Oracle HRMS.
- A **facility** is a healthcare institution, such as a hospital or a clinic.
- A **practice setting** is a sub-division of a facility that is classified as a clinical setting (for example, primary care clinic, rehabilitation, skilled nursing facility), in which patient care is delivered. A practice setting organization must be associated with a facility. When an organization unit has this organization role, care sites can be assigned to it.
- A **business unit** is a division or department of a healthcare facility or enterprise that supports an administrative, non-clinical function. It may correspond to a division or a department.

An organization unit can also have any of the following roles—in addition to or in lieu of the organization roles described above:

- *Medical Record Issuer* (`MRIssuer`) is an organization unit that can issue medical records.
- *Data controller* (`DataController`) is an organization unit that controls access to personal health information for patient consent management.
- *Order Filler* (`OrderFiller`) is an organization unit that can fill orders such as lab and nursing orders.

- *Custom Role* is any user-defined role. To create user-defined roles, extend the `CTB_ORG_ROLE` and `FND_ORG_CLASS` lookups.

  See Also:

  - *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page),* for seeded concept lists and values.
  - Table I–1 (Appendix I), for extensible concept lists that are the functional equivalent of FND lookups.

- *Healthcare Classification* is an organization unit with a generic healthcare role, which is used to identify HTB organization units and specify HTB organization unit attributes in Oracle HRMS. This role may also correspond to the Intermediate Node described in prior releases.

An organization role dictates its place within the enterprise hierarchy structure. Table 4–29 lists the possible valid parent/child organization role combinations:

Table 4–29 Parent/Child Organization Roles in Hierarchy

Parent Organization Role	Valid Child Organization Roles
Enterprise	■ Business Unit ■ Facility ■ Healthcare Classification
Facility	■ Business Unit ■ Facility ■ Healthcare Classification ■ Practice Setting
Business Unit	■ Business Unit ■ Facility ■ Healthcare Classification
Practice Setting	■ Practice Setting

Note: All organization units created through HTB APIs have an associated Healthcare Classification role. An organization unit with such a role can be placed anywhere within the enterprise hierarchy, as long as the restrictions for enterprise, facility, business unit, and practice setting are first applied.

4.3.2.2 Creating Organization Role Associations

Use the `createOrgRoleAssociation` method to associate a role with an organization. This method creates a subclassed role object that extends the base organization object.

Each subclassed role has its own specific attributes. For example, an organization with an Enterprise role has an Enterprise Type attribute; an organization with a Practice Setting role has Practice Setting Classifications and Practice Setting Features attributes.

Programmatic validation of parent/child compatibility and existence time low and existence time high applies when participating in a Healthcare Enterprise hierarchy. The base object can be further extended with custom (user-defined) roles.

4.3.2.3 Updating Organization Role Associations

Use the `updateOrgRoleAssociation` method to update a role for an organization unit.

4.3.2.4 Searching for Organization Roles

Use the `findOrgRoleAssociationsBySearchCriteria` method to find roles associated with an organization. This method extends the organization search criteria. It can search by both organization unit and organization role attributes, including the following:

- `Code`: The type of role, such as Enterprise, Facility, Business Unit, Practice Setting...
- `TypeCode`: The subtype of certain roles, such as Enterprise Type Code, Facility Type Code, Business Unit Type code, others...
- `Practice Setting Classification:` Classification codes and status.
- `Practice Setting Feature:` Feature codes and status.

Note: Organization role search criteria are defined by the `OrgRoleSearchCriteria` value object. See *Oracle Javadoc for HTB* for further details.

4.3.3 Implementing Organization Hierarchies

Reference

Oracle Javadoc for HTB

Table 4–30 summarizes Organization Hierarchy service and methods:

Table 4–30 Service and Methods: Organization Hierarchies

Level	Detail
Package	`oracle.apps.ctb.admin.org`
Class	`OrgHierarchyService`
Methods	▪ `createOrgHierarchy` ▪ `findOrgHierarchiesBySearchCriteria` ▪ `findOrgHierarchyElementsBySearchCriteria` ▪ `findOrgUnitsInHierarchy` ▪ `getOrgHierarchyById` ▪ `performAction` ▪ `terminateOrgHierarchy` ▪ `updateOrgHierarchy`

4.3.3.1 Creating Organization Hierarchies

Use the `createOrgHierarchy` method to create an instance of an organization hierarchy, based on the HR organization hierarchy model. The main hierarchy type is Healthcare Enterprise Hierarchy or `CTB_ENTERPRISE_HIER`. This hierarchy type is used to build an **enterprise** hierarchy, with specific business logic for parent/child relationships within the hierarchy. To create additional hierarchy types, extend the CTB_HIER_TYPE lookup. HTB hierarchy types can be associated with an existing HR hierarchy, making it sharable across multiple Oracle applications. Separate hierarchy objects, classified by one or more hierarchy types, can also be versioned, with support for multiple versions.

Note: Organization hierarchies are not versioned through the HTB Audit Service. They are modeled on Oracle HRMS hierarchy versioning functionality, where a new version can be created before changes are made to a hierarchy.

An enterprise organization is the root **node** of an enterprise hierarchy. The hierarchy is dynamic through time, meaning that its structure can change—by moving nodes within the hierarchy or removing organizations from the hierarchy. The hierarchy structure is always viewed or modified as of a specific version.

4.3.3.2 Updating Hierarchy Attributes

Use the `updateOrgHierarchy` method to update any attribute of an existing hierarchy object, with some restrictions on updating Healthcare Enterprise type hierarchies (an enterprise organization can only participate in one active enterprise hierarchy; the type cannot be removed, but can be inactivated).

You can also use this method to associate one or more HTB hierarchy types (including the seeded Healthcare Enterprise type or other user defined type) with an existing HR hierarchy.

Note: Hierarchy type is an HTB concept that lets HTB extend existing hierarchies; it is *not* a concept currently supported by HRMS.

4.3.3.3 Populating and Changing Hierarchy Structures

Use the `performAction` method to change hierarchy structures. This method lets you concurrently add organizations to a hierarchy, remove organizations from a hierarchy, and move organizations within a hierarchy. It creates a new version of the hierarchy (terminating the existing version) and accepts an array of add/move/remove actions, tracking all changes to the hierarchy structure by version number. Changes to the hierarchy structure are only performed from the latest hierarchy version.

This method does not let you update the version number or past versions; such updates are made using the `performAction` method. To update the latest version start date (as specified in the `performAction` method call that created the

version), use the `updateOrgHierarchy` method to update the version effective time low or effective time high.

See Also: Section 4.3.3.4, Terminating Organization Hierarchies

Notes:

- Set the Action array to `AddAction` to add organizations to a hierarchy.
- Set the Action array to `MoveAction` to move organizations within a hierarchy.
- Set the Action array to `RemoveAction` to remove organizations from a hierarchy.
- The Copy Flag parameter lets you copy the latest hierarchy structure or create a new hierarchy structure in the new version. This defaults to Yes.

- **Adding Organization Units to Hierarchies**

  Use the `performAction` method (Action array set to `AddAction`) to add an organization to an existing hierarchy; this action creates a new hierarchy version and terminates the old one. This method can be used when assigning free standing organizations to a hierarchy or for assigning hierarchically attached organizations to other hierarchies. *Note that an organization unit can only belong to one Enterprise type hierarchy at a time.*

  See Also: Section 4.3.1.1, Creating Organization Units

- **Moving Organization Units Within Hierarchies**

  Use the `performAction` method (Action array set to `MoveAction`) to move organization units and their children within a hierarchy; this action creates a new hierarchy version and terminates the old version. Moving organization units reflects business processes such as the transferring of practice settings to another facility. When moving an organization unit from one parent to another within the enterprise hierarchy, the validation rules relating to existence time low and existence time high and parent/child organization role compatibility apply equally as for adding organization units to an enterprise hierarchy. If the organization unit has children, the entire node of the hierarchy is moved to the new parent.

- **Removing Organization Units from Hierarchies**

  Use the `performAction` method (Action array set to `RemoveAction`) to remove organization units and their children from a hierarchy when they are no longer in use. This action creates a new hierarchy version and terminates the old version. Organizations that have children cannot be removed from the hierarchy. You must move or remove the children from the parent before removing the parent from the hierarchy.

4.3.3.4 Terminating Organization Hierarchies

Use the `terminateOrgHierarchy` method to terminate an organization hierarchy by setting the effective time high for the latest hierarchy version, without creating a new version—the latest version becomes the *terminal* version. A terminated hierarchy can subsequently be reactivated by using the `performAction` method to create a new version.

You cannot terminate enterprise type hierarchies, because this type of hierarchy is central to many HTB processes and may be referenced by other modules. If an enterprise type hierarchy is no longer valid, you can effectively supersede it by using the `performAction` method to create a new *version* of the hierarchy without copying the previous version, which is the default. You can then create organization unit relationships for this new version.

4.3.3.5 Searching and Retrieving Hierarchies

- **Searching for Hierarchies**

  Use the `findOrgHierarchiesBySearchCriteria` method to find hierarchy objects based on selected search criteria. This method also lets you search existing HRMS hierarchies that are not recognized by HTB (those that do not have an associated HTB hierarchy type).

- **Searching for Hierarchy Elements**

  Use the `findOrgHierarchyElementsBySearchCriteria` method to find hierarchy elements, including hierarchy relationships and a subset of organization attributes.

- **Searching for Organization Units**

  Use the `findOrgUnitsInHierarchy` method to find hierarchically assigned organization units within a hierarchy, by `SearchCriteria` attributes. Alternatively, you can search by other search criteria as well. See Table 4–30 and *Oracle Javadoc for HTB* for further details.

- **Retrieving Hierarchy Instance**

  Use the `getOrgHierarchyById` method to retrieve an instance of a hierarchy by its unique identifier.

4.4 Implementing Profile Option Services

Profile options are configurable preferences that affect the way an Oracle application looks and behaves. System administrators can control HTB behavior by setting profile option values. Application developers can control application behavior by programming their applications to perform in accordance with customized profile option values.

Examples of typical profile options include the following:

- **Language:** Determines the language in which the application is displayed to users.
- **Date Format:** Determines the format (mmddyyyy, ddmmyy...) for date displays.

System administrators can set profile options at the following five levels—listed from the highest level to the lowest:

- User
- Practice Setting
- Facility
- Enterprise
- Site

The profile option values set at each level define runtime values for each user's profile options. An option's runtime value is the highest level setting for that option.

When a profile option can be set at more than one level, an order of precedence applies: Site has the lowest priority, superseded by *Enterprise* which is superseded by *Facility*, which is superseded by *Practice Setting*, which is superseded by *User*. A profile option value entered at the Site level can thus be overridden by values entered at any other level, while a value entered at the User level has the highest priority.

For example, if the printer option is set only at the Facility and Practice Setting levels, the value set at the Practice Setting level applies, because it is the highest level setting for that option.

Seeded HTB profile options and default values are listed by Appendix D.

Prerequisites

The following tasks must be completed before setting profile options:

- **Implementing Organizations:** Organization Units of Enterprise, Facility and Practice Setting Organization Types must exist.
- **Implementing Security Services:** User Accounts must exist. The seeded responsibilities Healthcare Configuration Administrator and Healthcare Application Developer must also exist.

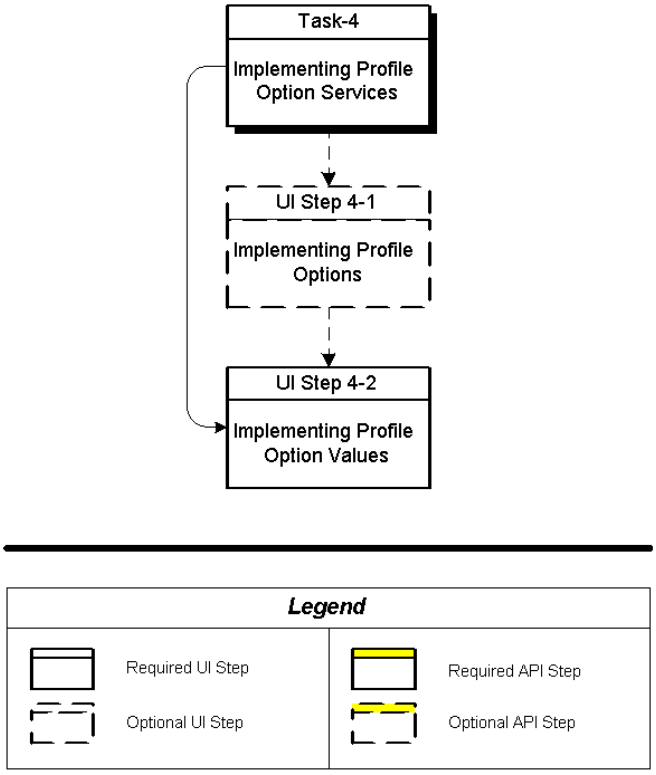
See Also:

- Section 4.3, **Implementing Organizations**
- Section 4.1, **Implementing Security Services**

Procedures

Figure 4–12 provides an overview of the implementation process for Profile Option Services:

Figure 4–12 Implementation Process: Profile Option Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–5, HTB Implementation Procedures: Profile Option Services

The following sections describe the implementation procedures for Profile Option Services (referenced by Figure 4–12):

- Figure 4.4.1, Implementing Profile Options
- Figure 4.4.2, Implementing Profile Option Values

4.4.1 Implementing Profile Options

Login

Log in to Oracle Applications using the user name you defined in Section 4.1.3.

Responsibility

Healthcare Application Developer

4.4.1.1 Creating Profile Options

The Healthcare Application Developer responsibility lets you create profile options using the window user interface. HTB applications can use the values defined for such profile options to customize behavior.

For example, a physician order entry application could define a profile option that defines the default sort order of patient problem lists (by date, severity,...).

Use this optional step to create custom profile options to drive customer application behavior.

Navigation

Table 4–31 summarizes the navigation paths used by this section:

Table 4–31 Navigation Paths: Creating Profile Options

Function or Window	Navigation Path
Create Profile Options Window	`Profile Options > Create Profile Option`

See Also: Appendix D, Seeded Profile Options

Steps

1. Complete the following sections of the Create Profile Option window:
   - **General Properties:** Complete the following fields:

Field	Description
Code	Unique code for the profile option.
Name	Name of the profile option.
Description	Description of the profile option.
Default Value	This value is returned if there is no value set at a level declared as required.
External Application Code	A reference to the application that uses the profile option.
Start Date	The first date the profile option is effective.
End Date	the date the profile option terminates.

- **System Administrator Access:** You can define if values for the profile option are (i) updatable by, or (ii) visible to the Healthcare Configuration Administrator responsibility—at each of the five available levels.

  For example, if System Administrator access at the User level is set to *Not Updatable but Visible*, the administrator can view the values set at the User level but cannot update or create them.

  See Also: Section 4.4.2, Implementing Profile Option Values

- **User Access:** Applications developed with HTB should use this setting to determine which user level profile option values are visible to, or can be updated by individual users.
- **Required Levels:** It is possible to optionally select any combination of the five profile option levels (User, Practice Setting, Facility, Enterprise, Site) as required levels. If a level is declared as *Required*, the profile option APIs do not inspect levels lower than the required level when returning values. They return either the value set at the required level or the default value—if a value is not set at the required level.
- **Value Type:** You can use the drop-down box to define one of the following value types, and you can also define certain constraints depending on the type:

Value Type	Description
Lookup Value	When you select this type, you must specify which lookup type (concept list) to validate against when a value is set. The window user interface signals an error condition if you attempt to set a value for a profile option that does not belong to the specified lookup type.
Number	The value must be a number; you can also specify a numeric range.
Text	The value must be text; you can specify a range for the text length.
Date	The value must be a valid date.

2. Click Apply to create the profile option; control returns to the Profile Options window, with the new profile option listed if it was successfully created.

4.4.1.2 Updating Profile Options

The Healthcare Application Developer responsibility lets you update profile options using the window user interface, including all properties of a profile option except its code.

Navigation

Click Profile Options in any Profile Options window to display currently defined profile options. Click the Update row-level icon to update the properties of a profile option.

Table 4–32 summarizes the navigation paths used by this section:

Table 4–32 Navigation Paths: Updating Profile Options

Function or Window	Navigation Path
Updating Profile Options	`Profile Options > Click update icon for the profile option to be modified`

See Also: Appendix D, Seeded Profile Options

Steps

1. You can modify the General Properties section of the Update Profile Option window:

Field Name	Description
Name	Change the name of the profile option.
Description	Change the description.
Default Value	Change the value returned if there is no value set at a required level.
External Application Code	Change the application reference.
Start Date	Change the effective date.
End Date	Change the termination date.
System Administrator Access	Change the updatable and visible settings for the healthcare Configuration Administrator responsibility—at each of the five available levels (User, Practice Setting, Facility, Enterprise).
User Access	Change the updatable and visible settings that determine if users of HTB-based applications can view or update their user level values for this profile option.
Required Levels	Change the required levels.
Value Type	Change the value type and corresponding constraints.

2. Click Apply to update the profile option; control returns to the Profile Options window.

4.4.2 Implementing Profile Option Values

Login

Log in to Oracle Applications using the user name you defined in Section 4.1.3.

Responsibility

Healthcare Configuration Administrator

4.4.2.1 Creating Profile Option Values

The Healthcare Configuration Administrator responsibility lets you create values for a profile option using the window user interface. System Administrator access settings for the profile option determine if this responsibility lets you create values at each of the five available levels.

See Also: Appendix D, Seeded Profile Options, for seeded profile options included with the HTB Platform.

Navigation

- Click System Profile Option values to display currently defined profile options:
- Click the View Profile Option Values icon to view the values of a profile option.
- Click Create Profile Option Value to create a new profile option value.
- Table 4–33 summarizes the navigation paths used by this section:

Table 4–33 Navigation Paths: Creating Profile Option Values

Function or Window	Navigation Path
Creating Profile Option Values	`Main Menu > System Profile Option Values > Click View Profile Option Values icon > Create Profile Option Value Button`
Viewing Profile Option Values	`Main Menu > System Profile Option Values`

See Also: Appendix D, Seeded Profile Options

Steps

1. Complete the following fields of the Create Profile Option Value window (Table 4–34):

Table 4–34 Create Profile Option Value Window Fields

Field Name	Description
Level	Select the level from the drop-down list; you can only select those levels that can be updated by a System Administrator.
Level Name	Select a specific instance of the chosen level using the Lookup function. The searchlight icon opens a lookup dialog for the chosen level. For example, if Facility is the chosen level, the lookup displays all of the facilities set up in HTB. Select one from the list. The Level Name field does not appear if the chosen level is *Site* (there is only one site in any installation).
Profile Option Value	Enter a value or select it from a lookup dialog—if the value type is specified as *Lookup Value* for this profile option.

2. Click Apply to create the profile option value. This returns control to the Profile Option Values window, with the newly created value listed if it was successfully created.

4.4.2.2 Updating Profile Option Values

The Healthcare Configuration Administrator responsibility lets you update values that have been set for a profile option using the window user interface. System Administrator access settings for the profile option determine if this responsibility lets you update values at each of the five available levels.

Navigation

- Click System Profile Option values to display currently defined profile options:
- Click the View Profile Option Values icon to view the values of a Profile Option.
- Click the Update icon to update an existing profile option value.
- Table 4–35 summarizes the navigation paths used by this section:

Table 4–35 Navigation Paths: Updating Profile Option Values

Function or Window	Navigation Path
Updating Profile Option Values	`Main Menu > System Profile Option Values > View Profile Option > Click Update icon`
Viewing Profile Option Values	`Main Menu > System Profile Option Values`

See Also: Appendix D, Seeded Profile Options

Steps

1. You can modify the following field in the Update Profile Option Value window:

Field Name	Description
Profile Option Value	Change the value directly, or click the searchlight icon to display the lookup dialog and select a value—if the value type is specified as *Lookup Value.*

2. Click Apply to submit the changes to the profile option value. This returns control to the Profile Option Values window, with the modified value listed if the update operation was successful.

4.4.2.3 Deleting Profile Option Values

The Healthcare Configuration Administrator responsibility lets you delete values that have been set for a profile option using the window user interface. System Administrator access settings for the profile option determine if this responsibility lets you delete values at each of the five available levels.

Navigation

- Click System Profile Option values to display currently defined profile options.
- Click the View Profile Option Values icon to view the values of a Profile Option.
- Table 4–36 summarizes the navigation paths used in this section:

Table 4–36 Navigation Paths: Deleting Profile Option Values

Function or Window	Navigation Path
Deleting Profile Option Values	`Main Menu > System Profile Option Values > Click View Profile Option Values > Click Delete icon`

See Also: Appendix D, Seeded Profile Options

Steps

1. You can delete existing values by clicking the row-level Delete icon.
2. If the deletion is successful, the window is refreshed with the deleted row omitted.

4.5 Implementing Enterprise Terminology Services

Enterprise Terminology Services (ETS) is a key component of Oracle Healthcare Transaction Base (HTB). It maintains coded vocabulary essential to HTB processing, providing support for:

- Concept-based terminology representation
- Intraterminology and interterminology equivalence
- Intraterminology mapping
- Concept Lists

ETS provides support for a generic terminology model, capable of hosting a variety of user terminology needs, and special support for a core set of terminologies that are standards in the industry. An API interface lets HTB-based applications access the features of the generic and **core terminologies**, thus enabling interoperability based on standard terminology.

Concept-based Terminology Representation

An ETS **coding scheme** is a structured system of terms or concepts used to maintain coded meanings. The *International Classification of Diseases, 9th Edition, Clinical Modification (ICD-9-CM)* and University Hospital's Laboratory Codes are two examples of **coding scheme**s.

Coding schemes represent individual concepts by using coded identifiers, or codes. The codes are alphabetic, numeric, or alphanumeric strings that uniquely identify a concept. The code assigned to a concept by the maintainer of its terminology is called its *concept code*.

Many coding schemes, or terminologies, are simple lists of descriptions and associated codes. Others are more complex, involving one or more of the following features, all of which are supported by ETS:

- Multiple text representations (also called descriptions, or labels) for a given concept, representing different languages or use cases.
- Concepts organized in hierarchical structures. The hierarchies may be single-valued hierarchies, where each concept sits in a single hierarchy (taxonomies), or multiple hierarchies, where a single concept may be the child of multiple parents.
- Other ontological relationships between concepts, such as *is caused by*, *predisposes to*, others...

ETS provides support for a special set of coding schemes, called Core ETS Terminologies, or core terminologies. The support takes the form of terminology-specific loaders and APIs. Currently supported core terminologies include the following:

- International Classification of Diseases, 9th Revision, Clinical Modification (ICD-9-CM)
- International Statistical Classification of Diseases and Related Health Problems, 10th Revision (ICD-10)
- SNOMED CT
- HCFA Common Procedure Coding System (HCPCS)(alpha-numeric codes, excluding the *D* dental codes)
- Logical Observation Identifier of Names and Codes (LOINC)
- Current Procedural Terminology (CPT4)
- Health Level 7 (HL7) version 3 vocabulary domains
- US Diagnosis Related Groups (DRG)
- US Major Diagnostic Categories (MDC)
- Portions of the FirstDataBank (FDB) US National Drug Data File (NDDF Plus)

ETS includes Java interface packages tailored to these core terminologies (example: an ICD-10 package, with classes such as `ICD10Concept`). The classes in these packages are extensions of the generic classes. Developers can access content from core terminologies using generic interfaces, and can also use the appropriate specialized classes to access additional, terminology-specific functionality.

A **coding scheme version** is a particular instance of a coding scheme. Examples include *ICD-9-CM for the year 2000*, *ICD-9-CM for the year 2001*, and the *University Hospital's Laboratory Codes, Updated September 2001*.

Each concept in each version can have several identifiers within ETS. Every concept has a **concept code**—the concept's code within its native terminology. For example, for the concept represented by *001* in coding scheme version ICD-9-CM v1 2003, *001* is its concept code within ETS.

The ETS concepts represented by *001* in version ICD-9-CM v1 2003 and *001* in version ICD-9-CM v1 2004 share the same concept code. However, ETS uniquely represents each concept internally, with its own internal identifier. The internal identifier for an ETS entity is generically called an ETSID; the ETSID for an ETS **concept** is called a **ConceptID**.

When a concept is added to a **concept list**, you assign an additional identifier to the concept, called a **membership code**, that uniquely identifies a concept within a concept list.

Interterminology Mapping

ETS supports **interterminology mapping** between different coding scheme versions (SNOMED CTv1.1, ICD-9-CM 2001,...) using the ETS cross map model. Each individual cross map represents a unidirectional mapping between a source concept and one or more target concepts. A cross map can be used to represent various types of relationships between concepts—a *broader-to-narrower* mapping, a *clinical-to-administrative* mapping, and so on. A map set contains all of the cross maps between concepts in two particular versions of the source and target coding schemes.

ETS specifies text file formats for map sets and cross maps. An ETS cross map loader accepts files in the specified formats and loads them into ETS tables. An ETS `CrossMap` interface lets applications navigate the loaded interterminology mapping content.

Intraterminology and Interterminology Equivalence

ETS manages **intraterminology equivalence** (information regarding concepts that are equivalent within the same coding scheme). For example, *001_CHOLERA* (from ICD-9-CM 2002) and *001 CHOLERA* (from ICD-9-CM 2003) are represented within ETS by two different internal identifiers. However, ETS gives these concepts the same meaning. Similarly, if in a particular version of a coding scheme, the editors of the coding scheme declare that concept A is retired and has the same meaning as active concept B, ETS can incorporate this information into its knowledge.

Where possible—when the terminology maintainers provide the equivalence information in a computable format and an ETS loader exists specifically for the terminology—ETS gathers this information at load time from the terminology files. When the terminology maintainers doe not provide such information, ETS depends upon change files that are loaded with a new terminology version. Change files identify code *re-uses* (codes that are now representing a different meaning than in a previous version) and reassignments (meanings that are now represented with different codes). Oracle provides change files for its core set of terminologies as well as other widely used terminologies. Customers can also implement their own local terminologies by creating change files for each new version and loading them using the ETS generic loader.

Interterminology equivalence incorporates the notion of semantic equivalence between concepts in different coding schemes. This information is loaded into ETS

using cross maps that have a special *Equivalence Context* designation. ETS loads these designated cross maps and makes them available to the cross map and equivalence interfaces.

The combination of **interterminology equivalence** and **intraterminology equivalence** makes it possible for a single interface call to retrieve concepts that are equivalence to a given concept, both within the concept's own terminology and within other terminologies.

Concept Lists

ETS also supports the use of **concept lists**—arbitrary lists of ETS concepts that can be used for a variety of purposes, including validation and user interface controls. Concept lists contain the valid sets of coded values used by HTB application program interfaces and by HTB Messaging Services.

HTB uses terminology codes in two areas:

- HTB Coded Attributes (using concept list functionality.
- HTB Master Catalog.

Terminologies must be set up in ETS to support these requirements.

See Also:

- Section 4.5.3, Creating and Loading ETS Cross Maps
- Section 4.5.6, Implementing Concept Lists
- Section 4.5.7, Implementing Interterminology and Intraterminology Equivalence
- Section 4.9, Implementing Clinical Business Services
- Section 4.9.1, Implementing Master Catalogs
- Appendix B, ETS Supported Terminologies and Cross Maps
- Appendix H, Empty Concept Lists
- *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page), for seeded concept lists and values.*

Prerequisites

- Implementing Organizations: Organization Units of Enterprise, Facility and Practice Setting Organization Types must exist.

- Implementing Security Services: User accounts must exist.

  See Also:

  - Section 4.1, Implementing Security Services
  - Section 4.3, Implementing Organizations

Login

Log in to Oracle Applications using the user name you defined in Section 4.1.3.

Responsibility

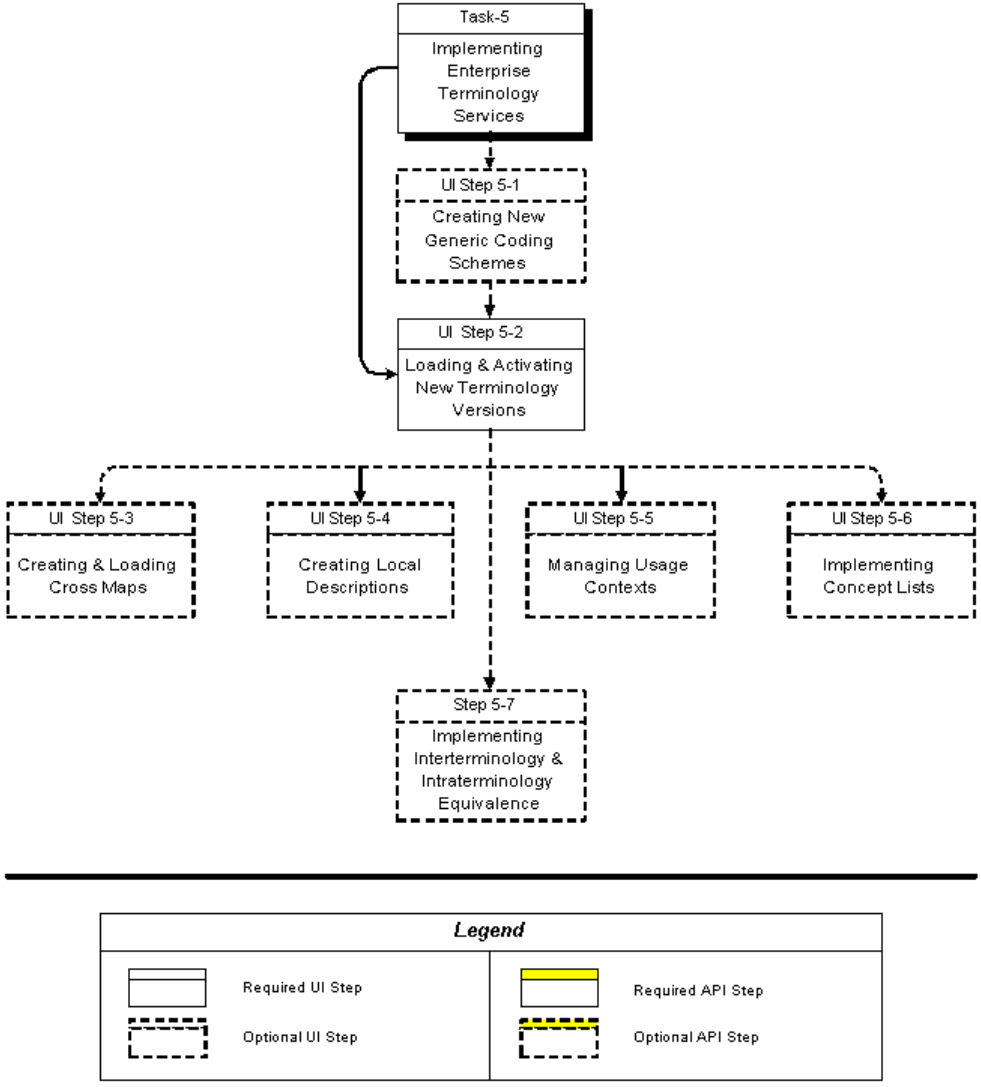
Healthcare ETS Administrator

Procedures

This section describes ETS implementation procedures, independent of the intended use of ETS constructs (for concept lists or the **master catalog**).

Figure 4–13 provides an overview of the implementation process for ETS:

Figure 4–13 Implementation Process: ETS



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–6, HTB Implementation Procedures: Enterprise Terminology Services (ETS)

The following sections describe the implementation procedures for ETS (referenced by Figure 4–13):

- Section 4.5.1, Creating New Generic Coding Schemes
- Section 4.5.2, Loading and Activating New Terminology Versions
- Section 4.5.3, Creating and Loading ETS Cross Maps
- Section 4.5.4, Creating Local Descriptions
- Section 4.5.5, Managing Usage Contexts
- Section 4.5.6, Implementing Concept Lists
- Section 4.5.7, Implementing Interterminology and Intraterminology Equivalence
- Section 4.5.8, Other ETS Support Functions

4.5.1 Creating New Generic Coding Schemes

Coding schemes must be created before they can be loaded. Coding schemes for the core set of ETS terminologies have been created prior to shipment for customer use. In addition, you can create and name generic coding schemes as needed.

Navigation

Table 4–37 summarizes the navigation paths used by this section:

Table 4–37 Navigation Paths: Creating Generic Coding Schemes

Function or Window	Navigation Path
Create Coding Scheme window	`Terminologies > Create Coding Scheme`

Steps

1. Navigate to the Create Coding Scheme window:
   - Click the Terminologies tab.
   - Click Create Coding Scheme.
2. Enter a name and description for the coding scheme in the text boxes.

   Note: The coding scheme name must be unique across coding schemes, and cannot be changed once created. Make the coding scheme name as descriptive as possible to avoid conflicts with other names subsequently created.

3. Click Apply to create a new generic coding scheme.

4.5.2 Loading and Activating New Terminology Versions

The process for loading and activating a new terminology version in ETS is the same for generic terminologies and for those in the ETS core set. Differences between loading a generic terminology version or a core set version only exist in the terminology file formats and the contents of the control file.

Note: *Do not load versions of terminologies that are seeded in HTB, including the following:*

- HL7
- IETF RFC 1766
- ISO 3166-1 alpha-2
- NUBC-UB92
- HTB Supplemental

If you have already loaded such versions, mark them as retired and non-default.

See Also: `HCT_TOP/patch/115/readme`, for details regarding file formats for ETS terminologies and loaders.

The process for loading and activating a new terminology version is also the same for the initial version of a coding scheme or for subsequent versions:

- Prepare the files.
- Load the version.
- Import the version.
- Activate the version.

There are two ways to load and import terminology versions in ETS: (i) using the ETS window user interface (which calls the Oracle Concurrent Manager), and (ii) using the Concurrent Manager and its interface directly. Both procedures are described below.

Notes:

- If you are loading a change file for a terminology version, the file must be included at the time the version is loaded—ETS does not support retrospective loading of change information.
- For equivalence processing to be performed correctly, versions must be loaded in order. Equivalence processing assumes that the codes referenced in the change file are from the version currently being loaded and its immediate predecessor.

See Also: `HCT_TOP/patch/115/readme/Change_File_Formats_General.txt`, for details about change files.

Navigation

Table 4–38 summarizes the navigation paths used by this section (applicable to the window user interface only):

Table 4–38 Navigation Paths: Loading and Activating New Terminology Versions

Function or Window	Navigation Path
Activate the version	`Terminologies > View Versions > Update > Apply`
Importing Terminologies	`Terminologies > View Versions > Publish`
Importing the staged version into the active space	`Terminologies > View Versions > Publish`

Table 4–38 (Cont.) Navigation Paths: Loading and Activating New Terminology

Function or Window	Navigation Path
Initiating concept import process	`Terminologies > Versions > Publish`
Load Version Window	`Terminologies > View Versions > Load Version`
Loading and Activating new terminology versions [Start load]	`Terminologies > View Versions > Load Staged Version > Apply`
Loading and Activating New Terminology Versions [Activate version]	`Terminologies > View Versions > Update > Apply`
Loading and Activating New Terminology Versions [Start import]	`Terminologies > View Versions > Publish`
Loading coding scheme versions	`Terminologies > View Versions > Load Staged Version`
Loading data and creating a new coding scheme version	`Terminologies > View Versions > Load Staged Version`
Terminologies Window	`Terminologies`
Update a published version	`Terminologies > View Versions > Update > Apply`
Update Published Version window	`Terminologies > Versions > Update Published Version`

Steps

Loading and Activating a Version

Using the ETS Window User Interface[1] for Loading and Importing:

1. Prepare the terminology content and control files (this step is the same regardless of the loading or importing method used).
   - If the version to be loaded is a generic terminology version of your own creation:
     - Create new terminology files in the format expected by the ETS generic loader.

[1] Calls the Oracle Concurrent Manager

See Also: `HCT_TOP/patch/115/readme`, for details regarding file formats for ETS terminologies and loaders.

- Move the terminology files into a directory located in the same file system as the Applications instance—a directory that is accessible by the Concurrent Manager.
- Create a control file that reflects the locations of the terminology files and move it into a directory located in the same file system as the Applications instance—a directory that is accessible by the Concurrent Manager.
- If the version to be loaded is a generic version obtained from a third party terminology support organization, adjust the files to correspond to the appropriate loader format.

See Also:

- `HCT_TOP/patch/115/readme`, for details regarding file formats for ETS terminologies and loaders.
- Oracle White Papers (available at *OracleMetaLink*) for a discussion of the generic model.

- Move the terminology files into a directory into a directory located in the same file system as the Applications instance—a directory that is accessible by the Concurrent Manager.
- Create a control file that reflects the locations of the terminology files and move it into a directory located in the same file system as the Applications instance—a directory that is accessible by the Concurrent Manager.
- Load data and create a new coding scheme version:

- Navigate to the Terminologies window by clicking the Terminologies tab.
- Select the coding scheme for the version to be created.

Examples:

- To load a version of an ETS core terminology such as SNOMED CT, locate SNOMED CT in the Coding Schemes table in the Terminologies window. Appendix B lists core terminologies that are supported by ETS. *Note that SNOMED CT is an extensive terminology, requiring significant processing time to load and import.*
- To load a user-defined generic terminology version, locate the user-defined name of the coding scheme.

- For the selected coding scheme, view its versions by clicking its View Versions icon.
- From the Versions window, load the new version:
  - Initiate the load process by selecting Load Staged Versions.
  - In the Load Version window, assign a name to the new version; *in future releases, the description text box will accept a description.*
  - Specify the location of the control file.

    See Also: `HCT_TOP/patch/115/readme`, for information about control files.

  - Click Apply. The browser returns to the Versions window. A confirmation banner displays the request ID of the ETS Terminology Loader request, in the Concurrent Manager schedule. Step 3 describes how to monitor the process.
  - When the loading process is complete, the new staged version is displayed in the Versions window. *Do not reload the Versions window to view the staged version—the browser Reload button is not supported in Oracle Applications*. To view the updated Versions window, navigate to the Terminologies window and select the appropriate coding scheme.

2. Monitor the progress of the concurrent process (optional step):
   - In a separate browser window, open another session as Healthcare ETS Administrator (Responsibility).
   - Under Concurrent Manager, click Monitor Requests.
   - In the Requests window:

- Select a Show parameter (All, Completed, Pending, Running, Search).
- Click Go.

  If Search was selected as the Show parameter, enter the search criteria and then click Go.

- In the new Requests window:

  Find the Request ID of the load request (in the rightmost column of the grid) to view the status of the ETS Terminology load or ETS Terminology import.

3. After completion of the load, import the coding scheme version:
   - Return to the Terminologies window, and view the coding scheme versions for which a new version is being created.
   - The recently loaded version appears in the Staged Version section of the window, with status *Loaded*. Import the staged version into the active space:
     - Initiate the import process by finding the recently loaded version in the Staged Versions area of the window and clicking its Publish icon. The browser updates the Versions window and a confirmation banner displays the request ID of the ETS Terminology Importer request, in the Concurrent Manager schedule.
     - You can monitor the process as described in (optional) step 2.
     - Upon completion of the import process, the new published version appears in the Versions window. *Do not reload the Versions window to view the published version—the browser Reload button is not supported in Oracle Applications.* To view the updated Versions Window, navigate to the Terminologies window and select the appropriate coding scheme.

     Note: In order to support concept equivalence, the ETS importer does not import a second *quarantined* version of a terminology if one already exists. This facilitates verification of equivalence between the quarantined version and the previous version of the terminology before the quarantined version is published.

4. Activate the version—this step is the same regardless of the loading or importing method used:

- Upon completion of the import process (publishing), the import version status is *quarantined*; activate the version and optionally set it as the default.

  Note: ETS equivalence processing requires that no more than one version of a coding scheme be quarantined at any one time. The importer refuses to import a second quarantined version of a coding scheme if one already exists.

- To activate the version, perform the procedure described by Section 4.5.8.2, applying the following information specific to this task:
  - When viewing the terminology versions, the recently published version appears in the Published Versions section of the Versions window; click its Update icon.
  - In the Update Published Version window, set the Status to Active to activate the version; you can also set the default status and description.

Using the Oracle Concurrent Manager directly for loading and importing:

1. Prepare the terminology content and control files (this step is the same regardless of the loading or importing method used).
   - If the version to be loaded is a generic terminology version of your own creation:
     - Create new terminology files in the format expected by the ETS generic loader.

       See Also:

       - `HCT_TOP/patch/115/readme`, for details regarding file formats for ETS terminologies and loaders.
       - Oracle White Papers (available at *OracleMetaLink*) for a discussion of the generic model.

     - Move the terminology files into a directory that is accessible by the Concurrent Manager.
     - Create a control file that reflects the locations of the terminology files and move the control file into a directory that is accessible by the Concurrent Manager.

- If the version to be loaded is a version obtained from a third party terminology maintenance organization, adjust them to correspond to the appropriate loader format.

See Also: `HCT_TOP/patch/115/readme`, for details regarding the formats for ETS terminologies and loaders.

- Move the terminology files into a directory that is accessible by the Concurrent Manager.
- Create a control file that reflects the locations of the terminology files and move the control file into a directory that is accessible by the Concurrent Manager.

2. Schedule the load manager to run:
   - Open a session using the Healthcare ETS Administrator responsibility.
   - Under the Concurrent Manager, click Schedule Requests. This launches a process that lets you schedule a job.
   - In the Name window:
     - The name of the program is chosen in the Schedule Request For field. Use the list of values to select the program called Healthcare ETS Terminology Loader.
     - Enter a name for the request in the Name field.
     - Click Next.
   - In the Parameters window:
     - In the Control File (absolute path) field, enter the path in which the control file is found.
     - In the Coding Scheme Name field, use the list of values to find and select the coding scheme name for which a version is to be loaded; enter the new version name.
     - Click Next.
   - Complete the scheduling by performing tasks in the Schedule, Notifications, and Printing windows (under normal circumstances, accept the defaults).

See Also:

- *Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests*
- Appendix M, Running Concurrent Programs

- In the Summary window, review your selections and click Submit.
- In the Information window, note the load sequence number (it will be used in step 4) and click OK.

3. Monitor the progress of the concurrent process:
   - In the Requests window:
     - Select a Show parameter (All, Completed, Pending, running, Search).
     - Click Go (if Search was selected as the Show parameter, enter search criteria and then click Go).
   - In the new Requests window:
     - Find the Request ID of the load request (step 2) to view the status of the ETS terminology load or ETS terminology report (shown in the Phase column).
     - Repeat the process until the phase shows *Completed*.
     - Upon completion, click the Details icon for the request; click View Log from the Details window.
     - Find the following line in the View Log and note the `LOADSEQ` (load sequence number)—it will be used in step 4:

       ```
       LOAD_HEADER_LOAD_SEQ (LOADSEQ=XXX)
       ```

4. After completion of the load, schedule the Import Manager to run:
   - Open a session using the Healthcare ETS Administrator responsibility.
   - Under the Concurrent Manager, click Schedule Requests. This launches a process that lets you schedule a job.
   - In the Name window:
     - The name of the program is chosen in the Schedule Request For field; use the list of values to select the program called Healthcare ETS Terminology Importer.

- Enter a name for the request in the Name field.
- Click Next.

- In the Parameters window:
  - In the Load Sequence Number field, enter the load sequence number.
  - In the Dry Run Mode field, select *Off* from the list of values.
  - Click Next.
- Complete the scheduling by performing tasks in the Schedule, Notifications, and Printing windows (under normal circumstances, accept the defaults).

  See Also:

  - *Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests*
  - Appendix M, Running Concurrent Programs
- In the Summary window, review your selections and click Submit.

  See Also: `HCT_TOP/patch/115/readme`, for details regarding file formats for ETS terminologies and loaders.

Note: In order to support concept equivalence, the ETS importer does not import a second *quarantined* version of a terminology if one already exists. This facilitates verification of equivalence between the quarantined version and the previous version of the terminology before the quarantined version is published.

---

5. Activate the version—this step is the same regardless of the loading or importing method used:
   - Upon completion of the import process (publishing), the import version status is *quarantined*; activate the version and optionally set it as the default.

Note: ETS equivalence processing requires that no more than one version of a coding scheme be quarantined at any one time. The importer refuses to import a second quarantined version of a coding scheme if one already exists.

- To activate the version, perform the procedure described by Section 4.5.8.2, applying the following information specific to this task:
  - When viewing the terminology versions, the recently published version appears in the Published Versions section of the Versions window; click its Update icon.
  - In the Update Published Version window, set the Status to Active to activate the version; you can also set the default status and description.

4.5.3 Creating and Loading ETS Cross Maps

Steps

1. Create a cross map file and a control file in ETS format, and move them into a directory located in the same file system as the Applications instance—a directory that is accessible by the Oracle Concurrent Manager.

   See Also:

   - Guidelines: Cross Maps
   - `HCT_TOP/patch/115/readme`, for details about Cross Map file formats.

2. Schedule the load manager to run:
   - Open a session using the Healthcare ETS Administrator responsibility.
   - Under Concurrent Manager, click Schedule Requests. This launches a process for scheduling a job.
   - In the Name window:
     - The name of the program is chosen in the Schedule Request For field. Use the list of values to select the program called *Healthcare ETS Terminology Loader.*
     - Enter a name for the request in the Name field.

- Click Next.

- In the Parameters window:
  - In the Control File (absolute path) field, enter the path in which the control file is found.
  - Use the list of values in the Coding Scheme Name field to find and select the coding scheme name called *Map Set Loader* (this is a dummy value ignored by the loader).
  - Enter text in the Coding Scheme Version Name field. The contents of this field are also ignored.
  - Click Next.
- Complete the scheduling by performing tasks in the Schedule, Notifications, and Printing windows (under normal circumstances, accept the defaults).

  See Also:

  - *Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests*
  - Appendix M, Running Concurrent Programs
- In the Summary window, review your selections and click Submit.
- In the Information window, note the load sequence number (it will be used in step 3) and click OK.

3. Monitor the progress of the concurrent process:
   - In the Requests window:
     - Select a Show parameter (All, Completed, Pending, Running, Search).
     - Click Go (if Search was selected as the Show parameter, enter the search criteria and click Go).
   - In the new Requests window:
     - Find the Request ID of the load request (step 2) to view the status of the ETS terminology load or ETS terminology import (shown in the Phase column).
     - Repeat the process until the phase shows *Completed*.

- Upon completion, click the Details icon for the request; click View Log from the Details window.
- Find the following line in the View Log and note the `LOADSEQ` (load sequence number)—it will be used in step 4:

  ```
  LOAD_HEADER_LOAD_SEQ (LOADSEQ=XXX)
  ```

4. After completion of the load, schedule the import manager to run:
   - Open a session using the Healthcare ETS Administrator responsibility.
   - Under Concurrent Manager, click Schedule Requests. This launches a process for scheduling a job.
   - In the Name window:
     - The name of the program is chosen in the Schedule Request For field. Use the list of values to select the program called *Healthcare ETS Terminology Importer.*
     - Enter a name for the request in the Name field.
     - Click Next.
   - In the Parameters window:
     - In the Load Sequence Number field, enter the load sequence number (step 3).
     - In the Dry Run Mode field, use the list of values to select *Off*.
     - Click Next.
   - Complete the scheduling by performing tasks in the Schedule, Notifications, and Printing windows (under normal circumstances, accept the defaults).

     See Also:

     - *Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests*
     - Appendix M, Running Concurrent Programs

   - In the Summary window, review your selections and click Submit.
   - In the Information window, note the load sequence number (to be used if the task progress is monitored) and click OK.

- You can monitor the task progress as described in step 3.

Guidelines

Cross Maps

The ETS Cross mapping model is based on the SNOMED CT cross mapping model. Cross-mapping mechanisms provide support for:

- Mapping a single concept to a target code (a one-to-one mapping).
- Mapping *to* a set of Target codes (a one-to-many mapping).

The current structure does *not* support:

- Mapping a set of Concepts to a Target.

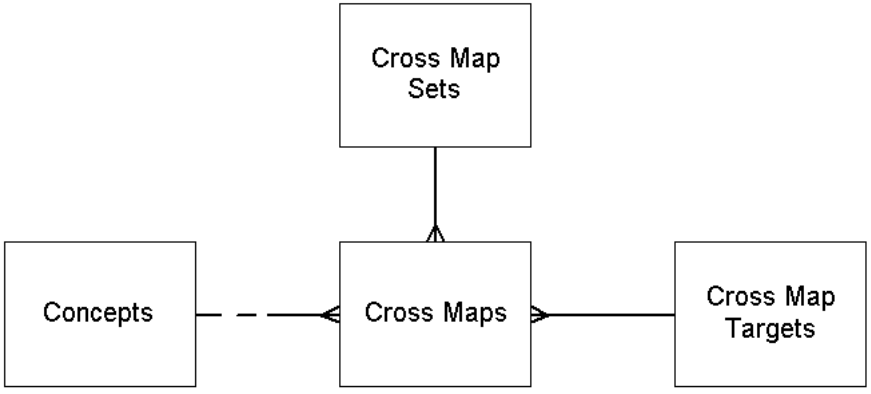
The ETS cross-map mechanism consists of the following tables:

- The Cross-Map Sets Table
- The Cross-Map Targets Table
- The Cross-Maps Table

> **See Also:** `HCT_TOP/patch/115/readme`, for the cross-map table structures.

The relationship between these tables is shown by Figure 4–14:

Figure 4–14 ETS Cross Mapping Relationship



A map set defines a mapping between two coding scheme versions, such as Terminology A and Terminology B. Each map set is composed of multiple cross maps. Each cross map consists of a source concept and one or more target concepts, such as a source concept from Terminology A and one or more target concepts from Terminology B—to which it maps.

Loading Cross Maps Provided by the College of American Pathologists[1]

The principal difference between cross map files distributed by the College of American Pathologists (with SNOMED CT) and those expected by ETS loaders is that the SNOMED CT files could contain data regarding multiple map sets in a single file. The map set file may contain multiple rows, each pertaining to a different map set. The cross map file may contain cross maps relating to multiple map sets, and the map targets file may contain targets used by multiple map sets (targets related to multiple coding schemes).

To make the SNOMED CT files suitable for ETS loading, split the files into map sets. The map set file should contain only one row, representing one map set. The cross maps file should contain only rows containing the map set ID of the chosen map set. The map targets file should contain only targets related to the target coding scheme specific to the map set.

4.5.4 Creating Local Descriptions

Navigation

Table 4–39 summarizes the navigation paths used by this section:

Table 4–39 Navigation Paths: Creating Local Descriptions

Function or Window	Navigation Path
Creating local concept description	`Terminologies > View Versions > View Concepts > Update > Create Local Description > Apply`
Create Local Description Window	`Terminologies > Versions > Concepts > Concept > Create Local Description`

Steps

1. Select the target concept and navigate to its Create Local Description window:

[1] SNOMED CT Cross Mappings

- Select a coding scheme from the Terminologies window and view its versions by clicking the View Versions link.
- Select a version from the Versions window and view its concepts by clicking the View Concepts link.
- Find the concept:
  - By paging through available concepts in the version (note: only the first 200 concepts in a terminology are returned), or
  - By searching for the concept.

  See Also: Section 4.5.8.1, Searching for a Concept in ETS.
- Navigate to the Concept window by clicking its Update icon.
- Navigate to the Create Local Description window by clicking Create Local Description.

2. Create the Local Description.
   - Provide a term (required).
   - Assign a preferred status (optional; defaults to No).
   - Assign a usage context (optional).
   - Click Apply to create the local description.

Guidelines

You can specify local descriptions for any ETS concept. These descriptions can be used by applications for display purposes, in place of terminology-specified descriptions. You can assign a single usage context to each local description.

See Also: Section 4.5.5, Managing Usage Contexts

The usage context for each local description must be unique; no two local concept descriptions can have the same usage context.

You can create local descriptions for retired or active concepts, but the typical procedure is to create a description for the concept in the active default version of the terminology.

4.5.5 Managing Usage Contexts

This section describes several tasks relating to usage contexts:

- Section 4.5.5.1, Creating a Usage Context
- Section 4.5.5.2, Deleting a Usage Context
- Section 4.5.5.3, Assigning a Usage Context to a Local Description
- Section 4.5.5.4, Associating a Usage Context with an Organization
- Section 4.5.5.5, Associating a Usage Context with a Concept List

4.5.5.1 Creating a Usage Context

Navigation

Table 4–40 summarizes the navigation paths used by this section:

Table 4–40 Navigation Paths: Creating a Usage Context

Function or Window	Navigation Path
Creating a usage context	`Usage Contexts > Create Usage Context`
Create Usage Context Window	`Usage Contexts > Create Usage Context`

Steps

1. Navigate to the Create Usage Context window.
   - Click the Usage Contexts tab.
   - In the Usage Contexts window, click *Create Usage Context*.
2. Assign a name and description to the new usage context.
3. Click Apply to create the new usage context.

See Also: Guidelines

4.5.5.2 Deleting a Usage Context

Navigation

Table 4–41 summarizes the navigation paths used by this section:

Table 4–41 Navigation Paths: Deleting a Usage Context

Function or Window	Navigation Path
Deleting a usage context	`Usage Contexts > Delete`

Steps

1. Navigate to the Usage Context window by clicking the Usage Contexts tab.
2. Find the usage context to be deleted, select it using the Select box, and click Delete.
3. Click Yes at the Warning window to delete the usage context.

4.5.5.3 Assigning a Usage Context to a Local Description

Navigation

Table 4–42 summarizes the navigation paths used by this section:

Table 4–42 Navigation Paths: Assigning a Usage Context to a Local Description

Function or Window	Navigation Path
Assigning usage context to a local description (during creation of a local description from the Concept Update window)	`Terminologies > View Versions > View Concepts > Update > Create Local Description > Apply`
Assigning usage context to a local description (during creation of a local description from the Update Membership Properties window)	`Concept Lists > View concepts > Update Membership Properties > Create Local Description > Apply`
Assigning usage context to a local description (existing local description)	`Terminologies > View Versions > View Concepts > Update > Update > Apply`
Create Local Description window (from Concept Lists)	`Concept Lists > View Concepts > Update Membership Properties > Create Local Description`
Create Local Description window (from Terminologies)	`Terminologies > Versions > Concepts > Concept > Create Local Description`
Update Local Description window	`Terminologies > Versions > Concepts > Concept > Update Local`

Steps

Assigning a Usage Context During Creation of a Local Description

If you are assigning a usage context during the process of creating the local description, see Section 4.5.4, Creating Local Descriptions.

Assigning a Usage Context to an Existing Local Description

1. If you are assigning a usage context to an existing local description, navigate to the concept and its local description:
   - Select the coding scheme from the Terminologies window and view its versions by clicking its View Versions link.
   - Select the appropriate version from the Versions window and view its Concepts by clicking its View Concepts link.
   - Find the concept by paging through available concepts in the version (only the first 200 concepts in a terminology are returned), or by searching for the concept.

     See Also: Section 4.5.8.1, Searching for a Concept in ETS

   - Click the concept's Update icon. In the Concept window, the local description is shown under Local Descriptions.
2. Click the Update icon of the target description.
3. In the Update Local Description window, assign or change the usage context and click Apply.

   Note: If another description for the concept is associated with the selected usage context, the existing association is dropped in favor of the new association.

4.5.5.4 Associating a Usage Context with an Organization

Steps

A usage context can be associated with an organization, using the method:

`associateUsageContextWithExternalOwner`

The usage context can subsequently be associated with a concept list (Section 4.5.5.5), resulting in a concept list-organization association, or with a local description (step 3, Section 4.5.5.3), resulting in a local description-organization association.

See Also: *Oracle Javadoc for HTB*

4.5.5.5 Associating a Usage Context with a Concept List

A usage context can be associated with a concept list at the time of concept list creation, or subsequently, by updating the concept list properties. Both processes are described in the following section (Steps).

Navigation

Table 4–43 summarizes the navigation paths used by this section:

Table 4–43 Navigation Paths: Associating a Usage Context with a Concept List

Function or Window	Navigation Path
Associate a usage context with a concept list at concept list creation	`Concept Lists > Create Concept List > Apply`
Associate a usage context with an existing concept list	`Concept Lists > Update Properties > Apply`
Create Concept List window	`Concept Lists > Create Concept List`
Update Concept List window	`Concept Lists > Update Concept List`

Steps

Associating a Usage Context at the time of Concept List Creation

1. Navigate to the Create Concept List window:
   - Click the Concept Lists tab.
   - In the Concept Lists window, select Create Concept List.
2. In the Create Concept list window:
   - Set the properties of the concept list.

     See Also: Section 4.5.6.1, Creating a Concept List

   - Select a usage context from the drop-down list.

3. Click Apply.

Associating the Usage Context by Updating the Concept List Properties

1. Navigate to the Update concept List window.
   - From either the Concept Lists window or a Specializations window, select a concept list and click its *Update Properties* icon.
2. In the Update Concept List window, select a usage context from the drop-down list.
3. Click Apply.

Guidelines

A usage context is used by applications to control the display of ETS concept local descriptions. When local descriptions are created (Section 4.5.4), they can be assigned a usage context. That local description is displayed by an application whenever the application specifies use of its usage context.

For example, the Utilization Review department might be accustomed to using short names or abbreviations for various diagnoses. Accordingly, a usage context called *Utilization Review* might be created. Local Descriptions can then be created for the appropriate concepts, and assigned the usage context of *Utilization Review*. Subsequently, applications the department uses to display or report information can use ETS APIs to present descriptions with the usage context of Utilization Review.

A unique, single usage context is permitted for each local description; multiple concept local descriptions cannot have the same usage context. Accordingly, if a local description is assigned a usage context and a local description for the concept already exists with that usage context, the operation succeeds—but the usage context is removed from the first local description.

A concept list associated with an organization (through a usage context) can similarly be used by an application. For example, the enterprise might have a concept list of medical services, entitled *ENT_MED_SERVICES*. Fair Oaks Community Hospital may require a specialization of that concept list that only contains a subset of the enterprise values. A usage context called *Fair Oaks* can be created and associated with Fair Oaks Community Hospital. A specialization of the `ENT_MED_SERVICES` concept list can subsequently be created, named `FAIR_OAKS_MED_SERVICES`, and associated with the Fair Oaks usage context. The following method lets you use the concept list specialization associated with a specific organization:

`getConceptListIdForOrganization`

4.5.6 Implementing Concept Lists

Concept lists provide a mechanism to group ETS concepts for a variety of purposes, such as user interface drop-down lists and other controls, or constraining values of an attribute to a certain set of coded values.

Concept lists provide additional features useful to applications. Concept list member concepts can possess activation and retirement dates; members can have active, retired, or pending statuses within a list. Each member concept in a concept list has a code by which it is known in the list. The code can be used by an application. It is unique among active or pending members of the concept list.

This section describes several tasks related to concept lists:

- Section 4.5.6.1, Creating a Concept List
- Section 4.5.6.2, Adding Concepts to a Concept List
- Section 4.5.6.3, Updating Concept List Properties
- Section 4.5.6.4, Updating Concept List Member Properties
- Section 4.5.6.5, Specializing a Concept List
- Section 4.5.6.6, Subsetting a Concept List
- Section 4.5.6.7, Migrating to New RIM Versions

HTB is shipped with a set of pre-defined concept lists. These concept lists are used within HTB to validate coded values in APIs and message processing:

- *Do not use these lists for other purposes.*
- The names of the predefined concept lists begin with `CTB_`—*do not use this prefix for concept lists that you create.*

Certain concept lists have been seeded empty—the concept list has been created before shipment, but no concepts have been added to it. These lists are shown in Appendix H —they must be filled by concepts during implementation. Sections 4.5.7.2 and 4.5.7.3 provide implementation instructions.

Note that you can add concepts to any seeded list that is defined as `EXTENSIBLE`, and you can create specializations of concept lists. Specializations are child concept lists that initially inherit parent concepts. In addition, specializations can possess the following inheritance types that permit various types of synchronization with concepts in the parent list:

- **Restricted:** Only concepts that are active in the parent list can be added to the child list.

- **Addition:** Concepts added to the parent list are added to the child list.
- **Deletion:** Concepts retired in the parent list are retired in the child list.

See Also:

- *Oracle Javadoc for HTB*
- *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page)*

A Specialization can be associated with a usage context, which may in turn be associated with a particular HTB organization. Section 4.5.6.5 provides specialization instructions.

You can also subset seeded lists, if only a subset of the seeded values are applicable. Because HTB seeded lists are of the type `SYSTEM` or `EXTENSIBLE`, and only lists designated `USER` can be subsetted directly, you must employ indirect methods to subset seeded lists. Section 4.5.6.6 provides subsetting instructions.

4.5.6.1 Creating a Concept List

Navigation

Table 4–44 summarizes the navigation paths used by this section:

Table 4–44 Navigation Paths: Creating a Concept List

Function or Window	Navigation Path
Creating a concept list	`Concept Lists > Create Concept List > Apply`
Create Concept List window	`Concept Lists > Create Concept List`

Steps

1. Navigate to the Create Concept List window:
   - Click the Concept Lists tab.
   - In the Concept Lists window, select Create Concept List.
2. In the Create Concept List window:
   - Enter a name for the concept list. The name must be unique across lists, and will be permanent (unchangeable).

- Enter a description for the list.
- Enter a status for the list (Active or Retired).
- Enter a group name for the list:
  - A group name serves to categorize the proposed use of the list
  - You can select an existing group name from the drop-down list.
  - Alternatively, you can enter a new group name by entering text in the text box.
- Enter inheritance information for the concept list:
  - The list can be a top-level list (no parent list), or the list can be a child, or specialization.
  - If a specialization, (i) select the name of the parent list from the list of values, and (ii) specify the inheritance types the specialization should possess (optional).
- Specify a usage context by selecting from the drop-down list (optional).

3. Click Apply to create a **concept list**.

4.5.6.2 Adding Concepts to a Concept List

Concepts can be added to any user-created concept list (those with extensibility *User*).

Concepts can also be added to certain lists that have been predefined for use by the applications programming interface (those with extensibility *Extensible*). Certain extensible predefined lists (Appendix H) have been shipped without seeded values, and must be added before using their respective functionality.

See Also:

- *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page)*
- Appendix H, Empty Concept Lists
- Appendix I, Concept List Equivalents

This section assumes that the concepts to be added to the list are contained in a terminology version already loaded into ETS. If you are not sure whether or not a concept list already exists in ETS, follow the steps in Section 4.5.8.1.

Note: Searching for a concept is coding scheme version-specific; you must search for a concept in each of the coding scheme versions that are likely to contain the target concept.

Once you have determined whether or not the concept to be added already exists in ETS (and its location), apply the following (Table 4–45);

Table 4–45 Procedure: Adding Concepts to Concept Lists

New Concept Exists In:	Coding Scheme Does Not Exist in ETS	Coding Scheme Exists in ETS
An existing version of an ETS terminology (core or generic)	NA	▪ Add concept according to the steps in this Implementation Guide Section.
A terminology version where: ▪ The terminology is not in the ETS core set, and ▪ The version has not been loaded in ETS *(including the case where user adds the desired concept to a new version of an existing user-maintained terminology)*	1. Format the new version as a generic terminology[1]. 2. Create a new coding scheme in ETS[2]. 3. Load and activate a new version[3]. 4. Add concept in accordance with this Implementation Guide Section.	1. Format the new version as a generic terminology[1]. 2. Load and activate a new version[3]. 3. Add concept in accordance with this Implementation Guide Section.
A terminology version where: ▪ The terminology is in the ETS core set, and ▪ The version has not been loaded into ETS.	NA—all terminologies in the ETS core set correspond to predefined ETS coding schemes.	1. Load and activate a new version[3]. 2. Add concept in accordance with this Implementation Guide Section.
No terminology.	1. Create a new terminology file, formatted as a generic terminology[1]. 2. Create a new coding scheme in ETS[2]. 3. Load and activate a new version[3]. 4. Add concept in accordance with this Implementation Guide Section.	NA

[1] See: Oracle White Papers (available at *OracleMetaLink*); Guidelines in this Implementation Guide Section; `HCT_TOP/patch/115/readme/`, for details regarding generic file formats, the ETS generic model, and manipulating files into the model.

[2] See: Section 4.5.1, Creating New Generic Coding Schemes.

[3] See: Section 4.5.2, Loading and Activating New Terminology Versions.

Navigation

Table 4–46 summarizes the navigation paths used by this section:

Table 4–46 Navigation Paths: Adding Concepts to a Concept List

Function or Window	Navigation Path
Add to concept list train (from viewing Coding Scheme Versions)	`Terminologies > View Versions > View Concepts > Add to Concept List train`
Add to concept list train (from viewing concepts in another concept list)	`Concept Lists > View Concepts > Add to Concept List train`
Add to concept list train (from viewing Related Concepts)	`Terminologies > View Versions > View Concepts > View Related Concepts > Add to Concept List train`
Adding concepts to a concept list (from viewing concepts in another concept list)	`Concept Lists > View Concepts > Add to Another Concept List > Next > Next > Next > Finish`
Adding concepts to a concept list (from viewing Related Concepts)	`Terminologies > View Viersions > View Concepts > Go > View Related Concepts > Add to Concept List > Next > Next > Next > Finish`
Adding concepts to a concept list (from viewing search results or coding scheme versions)	`Terminologies > View Versions > View Concepts > Go > Add to Concept List > Next > Next > Next > Finish`

Steps

1. Select the concepts:
   - Generate a display of the concepts to be added. The displayed concepts can be any of the following:
     - The result of a Search (Section 4.5.8.1):
     - The result of viewing concepts in a **coding scheme version**.
     - The result of viewing related concepts.
     - The result of viewing concepts in another concept list .
   - Select the concepts from the display.

2. Launch the *Add to Concept List* train by clicking *Add to Another Concept List* or *Add to Concept List*.
3. Proceed with the *Add to Concept List* train:
   - In the List Selection window, select the target concept list by one of the following methods:

Method-1: Entering Exact Name: ■

- Enter an *exact name* in the concept list field and select Next; click the flashlight icon.
- Search the list that follows by concept list name or description, by entering a string and standard SQL search characters and clicking Go.
- Select a concept list from the Results section; the name of the selected list appears in the Concept List field in the Select List page.
- Select Next.

Method-2: Entering Partial Name:

- Enter a *partial name* with a SQL search character in the Concept List field; click the flashlight icon.
- Select a concept list from the list that follows.
- Alternatively, you can perform a new search by concept list name or description and select a concept list from the results; the name of the selected list appears in the Concept List field in the Select List page.
- Select Next.

Notes:

- Concept list names and descriptions are not indexed via Oracle Intermedia (as are concept descriptions). Use typical SQL search patterns when searching for a concept list by name or description. In contrast, see: Section 4.5.8.1, Searching for a Concept in ETS.
- A concept cannot be added under the following conditions: (i) the concept is already active or pending in the selected list; (ii) the selected list is the parent of an additive child specialization and the concept is already active or pending on the child; (iii) the selected list is a SYSTEM list; or (iv) the selected list is a restricted child specialization and the concept is not active or pending on the parent lists.
- If any (but not all) concepts chosen for addition are not addable, a warning dialog lists the concepts that will not be added. You can elect to continue or return to list selection.
- If all concepts selected for addition are not addable, the List Selection window is reloaded with an information box listing these concepts. To continue, select another list.

- In the Membership Properties window, for each concept to be added, do the following:
  - Set the **membership code**: This string must be unique across all active or pending members of a concept list; it cannot be subsequently modified.

Note: *We strongly recommend that you use* `USER-` *as a prefix for any membership codes you add to Oracle-defined concept lists.* Other codes may be overwritten by Oracle-created content in subsequent releases.

  - Set the activation date and time for the concept (optional); the format is `DD-MMM-YYYY HH:MM:SS`, assuming the time zone of the installation. The entered activation date and time must be later than the current date and time plus one hour. A blank box defaults to the current date and time.

– Click Next.

Note: The Set Membership Properties page is reloaded with an error in any of the following conditions:

- You have entered the same membership code for multiple concepts.
- You have entered a membership code that is already associated with an active or pending concept on the list being updated.
- Activation dates precede the current time plus one hour.
- The list being updated is a restricted list and the activation date entered is earlier than the activation date of the corresponding concept in the parent list.

If any of these conditions occur, select a new membership code or activation date as appropriate, and click Next. Alternatively, click Back to return to the List Selection page, or click Cancel to exist the addition process.

- In the Review window, verify the concepts to be added, including their properties.
- Click Finish to complete the process. Following a confirming dialog, you are returned to the window viewed prior to launching the Add to Concept List train.

Note:

- If the concept is being added to a restricted specialization, and the selected activation date would cause the concept's active period to exceed that of the corresponding concept in the parent list, an exception occurs; a dialog appears, warning you that the concept has not been added.
- See Also: *Oracle Javadoc for HTB*

Caution:

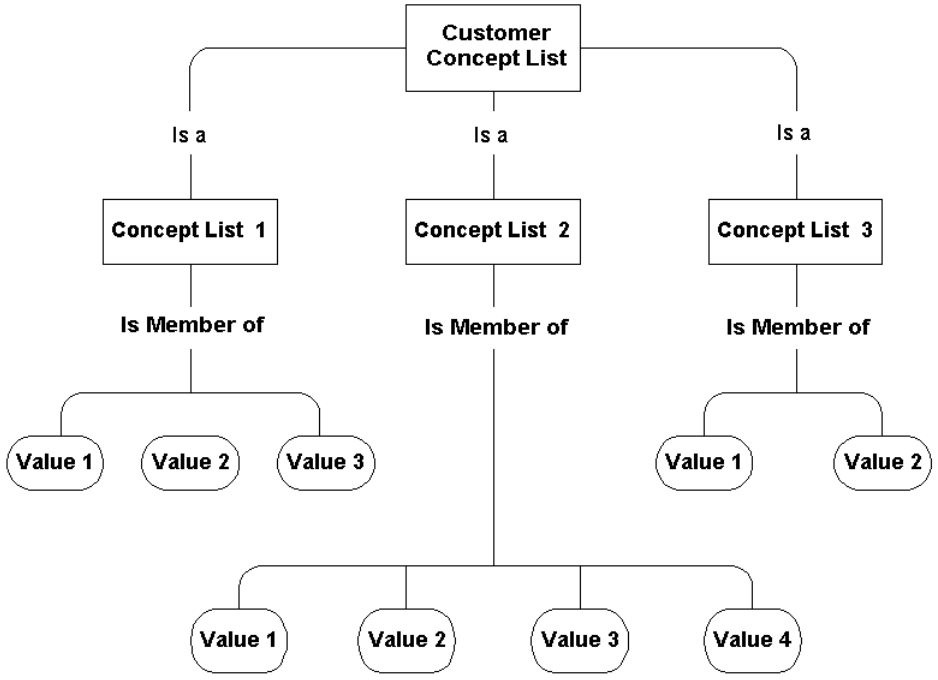
We strongly recommend that wherever possible, you only add concepts from the same terminology to a single concept list. Concept meanings can be sensitive to the context in which they are included in a terminology; mixing them with concepts from other terminologies may distort those meanings.

Guidelines

Recommendations for Creating Concept List Concepts

We recommend that you load all concepts that are not part of standard terminologies and will be used in your enterprise concept lists into a single generic terminology—for ease of maintenance, and to facilitate future migration. If you are loading the generic terminology into ETS for the first time, you can create a new generic terminology (Section 4.5.1). Figure 4–15 displays a conceptual model for a generic terminology for concept list concepts:

Figure 4–15 Conceptual View of Generic Terminology for Concept List Concepts



You can load the concepts into the three files of the generic full model—the Concepts file, the Descriptions file, and the Relationships file. The associated file content and structures are listed by the following tables:

- Table 4–47, Concepts File Content
- Table 4–48, Descriptions File Content
- Table 4–49, Relationships File Example

Table 4–47 Concepts File Content

Concept Code	Relationship Type Flag	Extended Type Code	Status Flag	Date Retired	Reassigned To	Attribute Name	Attribute Value
001	N	ROOT	A	...	...	Compo-nent	N

Table 4–47 (Cont.) Concepts File Content

Concept Code	Relationship Type Flag	Extended Type Code	Status Flag	Date Retired	Reassigned To	Attribute Name	Attribute Value
002	N	...	A	...	...	Compo-nent	Y
003	N	...	A	...	...	Compo-nent	Y
004	N	...	A	...	...	Compo-nent	Y
005	N	...	A	...	...	Compo-nent	Y
006	N	...	A	...	...	Compo-nent	Y
007	N	...	A	...	...	Compo-nent	Y
008	N	...	A	...	...	Compo-nent	Y
009	N	...	A	...	...	Compo-nent	Y
010	N	...	A	...	...	Compo-nent	Y
012	N	...	A	...	...	Compo-nent	Y
013	N	...	A	...	...	Compo-nent	Y
100	N	...	A	...	...	...	Y

Table 4–48 Descriptions File Content

Concept Code	Preferred Flag	Extended Type Code	Description Text	Status Flag	Date Retired	Attribute Name
001	Y	...	Customer X Lookups	A	...	...
002	Y	...	Concept List 1	A	...	...
003	Y	...	Concept List 2	A	...	...
004	Y	...	Concept List 3	A	...	...
005	Y	...	List 1 Value 1	A	...	...
006	Y	...	List 1 Value 2	A	...	...
007	Y	...	List 1 Value 3	A	...	...
008	Y	...	List 2 Value 1	A	...	...

Table 4–48 (Cont.) Descriptions File Content

Concept Code	Preferred Flag	Extended Type Code	Description Text	Status Flag	Date Retired	Attribute Name
009	Y	...	List 2 Value 2	A	...	...
010	Y	...	List 2 Value 3	A	...	...
011	Y	...	List 2 Value 4	A	...	...
012	Y	...	List 3 Value 1	A	...	...
013	Y	...	List 3 Value 2	A	...	...
100	Y	LONG	is a component of	A	...	...

Table 4–49 Relationships File Example

Source Concept Code	Relationship Type Concept Code	Target Concept Code	Status Flag	Date Retired	Attribute Name
002	100	001	A	...	...
003	100	001	A	...	...
004	100	001	A	...	...
005	100	002	A	...	...
006	100	002	A	...	...
007	100	002	A	...	...
008	100	003	A	...	...
009	100	003	A	...	...
010	100	003	A	...	...
011	100	003	A	...	...
012	100	004	A	...	...
013	100	004	A	...	...

These files are loaded as a **coding scheme version**. The steps described by Section 4.5.6.1 and Section 4.5.6.2 are used to create each concept and add concepts from the newly created terminology to the concept lists.

Subsequent additions to the concept lists are performed by following the procedures of this section or Section 4.5.6.2, depending upon whether or not the concepts to be added already exist in a loaded ETS coding scheme version.

Note: *We strongly recommend that wherever possible, you only add concepts from the same terminology to a single concept list.* Concept meanings can be sensitive to the context in which they are included in a terminology; mixing them with concepts from other terminologies may distort those meanings.

4.5.6.3 Updating Concept List Properties

Navigation

Table 4–50 summarizes the navigation paths used by this section:

Table 4–50 Navigation Paths: Updating Concept List Properties

Function or Window	Navigation Path
Update Concept List window	`concept Lists > Update Concept List`
Updating Concept List Properties	`Concept Lists > (Specializations) > Update Properties > Apply`

Steps

1. From the Concept Lists window, locate the target concept list by one of the following methods:
   - If it is visible in the Results section, click its Update Properties icon.
   - Otherwise, perform a search by concept list name or description:
     - Select Name or Description in the search box.
     - Enter either (i) a term that exactly matches the name or description, or (ii) a term with a standard SQL pattern matching character.
     - Click Go.
     - Locate the target concept list in the Results section and click its Update Properties icon.
2. In the Update Concept List window, update the following:
   - Description (enter text in the text box).
   - Status (Active, Retired).
   - Inheritance Type (for specializations only; Additive and Subtractive types are updatable).

- Usage Context (select from entries in the drop-down list).

3. Click Apply to complete the process.

See Also: *Oracle Javadoc for HTB*

4.5.6.4 Updating Concept List Member Properties

Navigation

Table 4–51 summarizes the navigation paths used by this section:

Table 4–51 Navigation Paths: Updating Concept List Member Properties

Function or Window	Navigation Path
Update Concept List Member Properties window	`Concept Lists > (Specializations) > View Concepts > Update Membership Properties`
Updating Concept List Member Properties	`Concept Lists > (Specializations) > View Concepts > Update membership Properties > Apply`

Steps

1. From the View Concepts window for a concept list, select the concept whose properties are to be updated and click its *Update Membership Properties* icon.
2. In the Update Membership Properties window, update properties of the concept:
   - Core member setting: Indicates if the member should be included in the *core set* for this concept list. The members of the core set can be used by applications, for drop-down lists or other display controls.
   - Default setting: Indicates if the member is the default concept in the list. Only one concept can be designated as the default in a list. If it is set when another member is designated as the default, the other member loses default status.
   - Activation date and time: The date and time at which the member status should change to *Active*. The format is `DD-MMM-YYY HH:MM:SS`, assuming the time zone of the installation. A blank defaults to the current date and time.

Notes:

- The activation date of a concept in a restricted child list cannot be changed so as to cause the active period of the concept in the *child list* to grow beyond the active period of the concept in the *parent list.*
- Modifying the activation date of a concept in a list that is the parent of a restricted child list may affect the corresponding concept in the child list.
- See: *Oracle Javadoc for HTB* for further details.

- Retirement date: The date and time at which member status changes from *Active* to *Retired*. The retirement date does not have to be specified for an active concept.

Notes:

- The retirement date of a concept in a restricted *child list* cannot be changed so as to cause the active period of the concept in the child list to grow beyond the active period of the concept in the *parent list*.
- Modifying the retirement date of a concept in a list that is the parent of a restricted child list may affect the corresponding concept in the child list.
- Retiring concepts from the parent of a restricted or deletion inheritance child may affect the child list.
- See: *Oracle Javadoc for HTB* for further details.

3. Click Apply to complete the update.

4.5.6.5 Specializing a Concept List

Concept lists can be *specialized*. A specialization of a concept list is a child concept list that initially inherits the active members of the parent list. It is a separate concept list, distinct from the parent list. Subsequent behavior of the specialization (a child concept list) with respect to the parent concept list depends upon the setting of its inheritance type:

- Addition inheritance: Any concept added to the parent list is added to the child list.
- Deletion inheritance: Any concept retired from the parent list is retired from the child list.
- Restricted inheritance: A child list cannot contain any concept not contained in its parent list; before a concept is added to the child list it must first either exist in the parent list or be added to the parent list. A concept in a restricted child list also inherits certain changes to the activation and retirement dates of the corresponding concept in the parent list.

The inheritance types are not mutually exclusive. A restrictive list must also exhibit deletion inheritance. You can update a list's addition inheritance and deletion inheritance by turning them on or off, but you cannot update its restricted inheritance.

A specialization can be associated with a usage context, as can a concept local description. A usage context can in turn be associated with an organization. Accordingly, a concept list can have multiple specializations, each associated with a particular organization.

A specialization can be associated with a usage context (Section 4.5.5.5). A specialization can be associated with an organization by associating it with a usage context that has first been associated with an organization (Section 4.5.5.4). The specialization can then be retrieved using the method:

`getConceptListIdForOrganization`

A concept list specialization is created in the same manner as any other concept list (Steps).

Values in a concept list specialization can be added or retired as for any concept list, in accordance with Sections 4.5.6.2 and 4.5.6.6.

See Also: *Oracle Javadoc for HTB*

Navigation

Table 4–52 summarizes the navigation paths used by this section:

Table 4–52 Navigation Paths: Specializing a Concept List

Function or Window	Navigation Path
Create Concept List window	`Concept Lists > Create concept List`
Specializing a concept list	`Concept Lists > Create Concept List > Apply`

Steps

1. Click the Concept Lists tab and select Create Concept List.
2. In the Create Concept List window:
   - Enter a name for the concept list. The name must be unique across lists, and will be permanent (unchangeable).
   - Enter a description for the list.
   - Enter a status for the list (Active or Retired).
   - Enter a group name for the list:
     - A group name serves to categorize the proposed use of the list
     - You can select an existing group name from the drop-down list.
     - Alternatively, you can enter a new group name by entering text in the text box.
   - Enter inheritance information for the concept list:
     - Select the name of the parent list from the list of values.
     - Specify the inheritance types the specialization should possess (optional).
   - Specify a usage context by selecting from the drop-down list (optional).
3. Click Apply to create the specialization.

4.5.6.6 Subsetting a Concept List

It may be desirable to subset a concept list—using only a subset of a concept list's members, for UI display purposes, or for validating data to be stored by an application. The following sections describe how to subset a concept list:

- Subsetting a User Concept List
- Subsetting a Concept List of any Extensibility Type

Subsetting a User Concept List

A concept list of extensibility type *User* can be subsetted by retiring unwanted members from the list. A member is retired by updating its retirement date and time.

Note: This subsetting procedure does not apply to System or System extensible concept lists.

Navigation

Table 4–53 summarizes the navigation paths used by this section:

Table 4–53 Navigation Paths: Specializing a User Concept List

Function or Window	Navigation Path
Subsetting a user concept list	`Concept Lists > (Specializations) > View Concepts > Update Membership Properties > Apply`
Update Concept List Member Properties window	`Concept Lists > (Specializations) > Concepts > Update Membership Properties`

Steps

1. From the View Concepts window for a concept list, select the concept whose properties are to be updated and click its *Update Membership Properties* icon.
2. In the Update Membership Properties window, update the retirement date and time of the concept.

Notes:

- The retirement date of a concept in a restricted child list cannot be changed so as to cause the active period of the concept in the *child list* to grow beyond the active period of the concept in the *parent list.*
- Modifying the retirement date of a concept in a list that is the parent of a restricted child list may affect the corresponding concept in the child list.
- Retiring concepts from the parent of a restricted or deletion inheritance child may affect the child list.
- See: *Oracle Javadoc for HTB* for further details.

3. Click Apply to complete the update.

Subsetting a Concept List of any Extensibility Type

A concept list of any extensibility type (including *user*) can be subsetted using either of two additional procedures. These procedures are especially useful if the list to be subsetted is a *System* or *System Extensible* list, from which members cannot be retired:

Using the Core Member Setting of List Members

Navigation

Table 4–54 summarizes the navigation paths used by this section:

Table 4–54 Navigation Paths: Using Core Member Setting to Subset a Concept List

Function or Window	Navigation Path
Subsetting a concept list of any extensibility type: Using the core member setting of list members	`Concept Lists > (Specializations) > View Concepts > Update Membership Properties > Apply`
Update Concept List Member Properties window	`Concept Lists > (Specializations) > View Concepts > Update Membership Properties`

Steps

1. From the View Concepts window for a concept list, select a concept to be designated as a core member and click its *Update Membership Properties* icon.
2. In the Update Membership Properties window, set the core member property of the concept to *Yes*.
3. Click Apply to complete the update.
4. Repeat these steps for each member to be designated as a core member.

The core set of members in the list can then be retrieved using the method:

`getCoreSet`

Checks of individual members of the list can be performed using the method:

`isCoreMember`

See Also: *Oracle Javadoc for HTB*

Using a Specialization of the Concept List and Retiring Members

You can create a specialization of the concept list (Section 4.5.6.5), and specify a usage context for the child list. You can then subset the child concept list (Section 4.5.6.6), and you can use the subsetted list as required by your application.

Access the specialization using the method:

```
getChildconceptList
```

See Also: *Oracle Javadoc for HTB*

4.5.6.7 Migrating to New RIM Versions

New HTB releases conform to the latest HL7 RIM vocabulary version. As a consequence, a new release may implement concept lists that are different from those implemented in prior releases. The *Oracle About Doc* associated with each release defines the RIM version implemented with that release, and describes the concepts added, modified, or removed from the prior release.

When HTB is implemented, it overwrites concept lists in an existing implementation. While this is not an issue if you are implementing HTB for the first time, it requires intervention to migrate from the existing implementation of a prior release.

This section describes the additional steps you must perform to migrate ETS concepts from an existing implementation to those supported by a new release.

There are three different scenarios related to the migration of concept lists:

- A new concept is added.
- An existing concept is replaced with a semantically equivalent concept.
- A concept is removed.

Adding New Concepts

No additional implementation steps are required to use a new concept that did not exist in the prior release. Inbound messages can use new concepts without any additional implementation steps required. However, applications source code modifications may be required to implement special behavior based on the new concepts.

Replacing Existing Concepts

If an existing concept is replaced by a semantically equivalent concept, no additional implementation steps are required if an application utilizes ETS Concept

Equivalence services. The Inbound Message Processor uses concept equivalence to determine if an inbound concept is semantically equivalent to an existing member of a concept list.

Removing Concepts

If a concept is removed by the new release, additional implementation steps are required to continue using the removed concepts.

To continue using such concepts, either in inbound messages or directly in applications that use HTB APIs, perform the following steps:

1. Identify the removed concepts that are required by your application.
2. Locate each required concept in the ETS Workbench.

   See Also: Section 4.5.8.1 for information about locating concepts in ETS.

3. Add the removed concept to the relevant concept list, using the specified **membership code**.

   See Also: Section 4.5.6.2 for information about adding concepts to a concept list.

4.5.7 Implementing Interterminology and Intraterminology Equivalence

Coded information in HTB is represented by concepts that are drawn from different ETS terminologies and their versions. With the passage of time and the sharing of the repository by different applications, the same meaning may be recorded by different concepts. When a query uses one of these concepts as its parameter, it skips records that contain the same meaning encoded by a concept from a different version of the same terminology or from a different terminology altogether.

When two concepts—from the same or different terminologies—have the same meaning, they are *equivalent*. Concept equivalence facilitates the specification and query of equivalence between concepts in ETS. When combined with the kind of query described above, it lets you retrieve all records that contain concepts with the same meaning.

Concept equivalence is used by the Inbound Message Processor (IMP) to validate codes in inbound messages against concept lists and act definitions. The Outbound Message Processor (OMP) uses this service to translate codes from the repository into equivalent codes that are configured for different receivers.

Concept equivalence services in ETS include two categories:

- *Intra*terminology Equivalence
- *Inter*terminology Equivalence

*Intra*terminology Equivalence

Intraterminology equivalence deals with identical concepts (those with the same meaning) within a single terminology. When a new version of a terminology is released, there may be several changes to the representation and meaning of concepts when compared to the previous version. Because there is no way for ETS to automatically determine these changes, by default it treats concepts from the previous and new versions as distinct and unrelated. However, it is possible to explicitly indicate the changes that have occurred between a prior version and a new version in a **change file** that is loaded with the new version. Using this information, Intraterminology equivalence services can determine whether two concepts from the previous version and the new version have the same meaning.

Given a concept from a version of a terminology, ETS can retrieve equivalent concepts from all contiguous versions that have change files loaded. Given two concepts from different versions of a terminology, ETS can verify if they are equivalent, provided that the more recent version and all the intermediate versions have change files loaded.

*Inter*terminology Equivalence

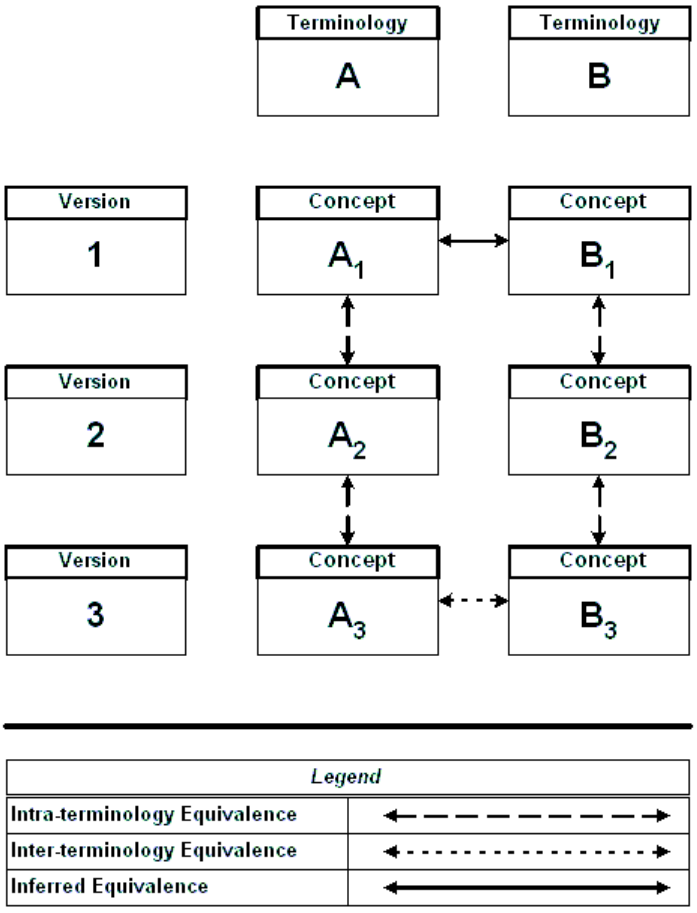
Interterminology equivalence deals with identical concepts (those with the same meaning) from different terminologies. Concepts from two different terminologies can vary widely in their granularity and coverage of a domain. Because there is no way for ETS to automatically determine these differences, by default it treats concepts from the two terminologies as distinct and unrelated. However, it is possible to explicitly indicate equivalence between concepts from two versions of different terminologies in the form of an ***Interterminology Mapping***. Using this information, Interterminology equivalence services can determine whether two concepts from different terminologies have the same meaning.

Combining *Intra*terminology and *Inter*terminology Equivalence

Equivalence between concepts is a transitive relationship. In Figure 4–16, if Concept A_1 is equivalent to Concept A_2, and Concept A_2 is equivalent to Concept A_3, it can be inferred that Concept A_1 is equivalent to Concept A_3. Consistent with this logic, Concept Equivalence services in ETS can determine if concepts from two terminologies are equivalent—provided that (i) an **interterminology mapping**

exists between versions of the two terminologies, and (ii) change files have been loaded for all versions.

Figure 4–16 Concept Equivalence Model



In Figure 4–16, ETS transitively combines intraterminology equivalence and interterminology equivalence information to infer that Concept A_1 is equivalent to Concept B_1.

Prerequisites

None

Login

Log in to Oracle Applications using the user name you defined in Section 4.1.3

Responsibility

Healthcare ETS Administrator

Procedures

The following sections describe implementation procedures for both intraterminology and interterminology equivalence:

- Section 4.5.7.1, Creating and Loading Change Files
- Section 4.5.7.2, Creating and Loading Interterminology Mappings with Equivalence Data
- Section 4.5.7.3, Invoking Concept Equivalence Services

4.5.7.1 Creating and Loading Change Files

Change files are used to document the differences between successive versions of a terminology that are loaded into ETS, in a format that is acceptable for loading purposes. Change files are loaded at the same time as the terminology version data using the same loader and importer. Change files contain the following types of information:

- **Reassignment:** The meaning of one concept is occasionally *reassigned* to another concept represented by a different concept code. Following are some of the situations in which reassignment occurs:
  - **Duplicate concepts are detected.** One of them is elected to continue representing the meaning while the other is retired or deleted and reassigned to the retained concept.
  - **A concept is detected to be erroneous.** The erroneous concept is retired or deleted and reassigned to a correct concept.
  - **The classification of a concept is changed**. If the concept codes are hierarchical (as in ICD-9-CM), the change in classification translates to a change in concept code, necessitating reassignment.

If both the source and target concepts of a reassignment are from the new version, the reassignment is said to be *intra-version*. For example, in SNOMED-CT, if a duplicate or erroneous concept is detected, the new version carries forward the duplicate or erroneous concept in an inactive status. The reassignment in this case is from the inactive concept in the new version to an active concept in the new version.

If the source of a reassignment is from the previous version and the target is from the new version, the reassignment is said to be *inter-version*. For example, in ICD-9-CM, if an erroneous concept is detected, it is deleted and excluded from the new version. A correct concept is provided in the new version and a reassignment is created between the concepts in the previous version and the new version.

- **Reuse:** Occasionally, a concept code used in the previous version is *reused* to represent a concept with a different meaning in the new version. Note that this is considered bad terminology practice and should only be used to account for inadvertent errors. ***Unless a reuse is explicitly called out in the change file, a concept in the previous version is always considered equivalent to a concept with the same concept code in the new version.***

Change files are preseeded for the following terminologies (no further implementation steps are required):

- HL7
- HTB Supplemental

For each of the following terminologies, new versions and their change files are available to customers possessing licenses from the vendor of the terminology; these must be loaded into ETS as described in Section 4.5.2:

- IETF RFC 1766
- ISO 3166-1 alpha2
- NUBC-UB92
- CPT-4
- FDB
- HCPCS-2
- ICD-10
- ICD-9-CM-DRG
- ICD-9-CM-MDC

- ICD-9-CM-V1
- ICD-9-CM-V3
- LOINC
- SNOMED-CT

For all other custom terminologies that are loaded into ETS, change files must be created separately and loaded for each new version as described in the implementation steps.

Note that change files must be loaded concurrently with the new version of a terminology. It is not possible to load a change file for a version of a terminology *after* both versions have been loaded.

Note also that a change file can only equivalence concepts between two consecutively loaded versions of a terminology. The following scenarios illustrate this constraint:

- A concept (concept code X) exists in version 1 of a coding scheme. The concept neither appears in version 2 nor is reassigned to an equivalent version 2 concept. A concept with code X reappears in version 3 with the same meaning as in version 1. It is not possible to indicate to ETS that concept X from version 1 is equivalent to concept X from version 3—because it spans a version.
- A concept (concept code X) exists in version 1 of a coding scheme. The concept neither appears in version 2 nor is reassigned to an equivalent version 2 concept. In version 3, another concept (concept code Y) is created that is equivalent to concept X from version 1. It is not possible to indicate to ETS that concept X from version 1 is equivalent to concept Y from version 3.

> **Note:** In order to support concept equivalence, the ETS importer will not import a second quarantined version of a terminology if one already exists. This facilitates verification of equivalence between the quarantined version and the previous version of the terminology before the quarantined version is published.

Steps

1. Determine whether the reassignment information for the terminology is Inter-version or Intra-version. Use the following rules to make this determination:

- Intra-version reassignment is used by terminologies that carry forward the duplicate or erroneous concept into the new version, albeit with a retired status, and reassign it to an active concept with a different concept code in the new version.
- Inter-version equivalence should be used for terminologies that do not carry over the concept to be reassigned into the new version. Such terminologies will instead reassign directly from the concept in the previous version to a concept in the new version.

2. Create the change file with the appropriate Reassignment and Reuse entries. See `HCT_TOP/patch/115/readme /Change_File_Formats_General.txt` for details about the format of the change file. Note that the **S** `ENTRY_TYPE` is called *Reassignment* in this guide.
3. Set the `HISTORY_TYPE` property of the terminology loader control file to `INTRAVERSION` or `INTERVERSION` based on the determination made in Step 1. If the `HISTORY_TYPE` is set to `INTRAVERSION`, both the `SOURCE_CONCEPT_CODE` and `TARGET_CONCEPT_CODE` of a reassignment are assumed to be from the version being loaded. If the `HISTORY_TYPE` is `INTERVERSION`, the `SOURCE_CONCEPT_CODE` is assumed to be from the previous version, and the `TARGET_CONCEPT_CODE` is assumed to be from the new version.

> **Note:** If no `HISTORY_TYPE` is present, the value defaults to `NONE` and no equivalence information is processed. If the `HISTORY_TYPE` is `INTERVERSION` or `INTRAVERSION`, a change file must be specified using the `CHANGE_FILE` property. Even if no reassignment or reuse has occurred, an empty change file with the header line must be provided.

4. Specify the location of the change file using the `CHANGE_FILE` property of the terminology loader control file.
5. Execute the steps specified in Section 4.5.2 to load and activate the terminology version along with the change file.

4.5.7.2 Creating and Loading Interterminology Mappings with Equivalence Data

Interterminology mappings provide a mechanism by which concepts from a source version in one terminology can be mapped to concepts from a target version in another terminology. Mappings are typically tailored for a specific application. For example, a data-aggregating or reporting application may require a mapping

between specialized SNOMED-CT codes and coarse ICD-9-CM codes. A data retrieval application may use mappings with the opposite semantics (from less granular classifier codes to more detailed codes). These examples serve to illustrate that mappings serve multiple purposes, and not all cross maps indicate equivalence. Those cross maps that truly do indicate equivalence must be explicitly flagged by the author of the cross maps. This section describes the steps you should follow to indicate equivalence between concepts from two different terminologies using Interterminology Mapping files.

Steps

1. Create Map Set Loader files (Section 4.5.3).
2. Perform the following steps on the file referenced by the `CROSS_MAP_FILE` property of the Map Set Loader Control File (see `HCT_TOP/patch/115/readme /Terminology_File_Formats_MapSet.txt` for Map Set Loader file formats).
   - The eighth column of the cross maps file is called `EQUIVALENCECONTEXT`. The Map Set Loader inspects this column in each row to determine if the cross map in that row can be used for equivalence. If this column in a particular row is left empty or set to null, the cross map in that row will not be used by ETS Concept Equivalence Services.
   - If the `EQUIVALENCECONTEXT` column is populated, the cross map is interpreted as indicating equivalence between the concepts represented by the `MAPCONCEPTID` and the `MAPTARGETID`.
   - Because determining similarity of meaning between concepts from different terminologies is often subjective, it may not be appropriate to use the same set of cross maps for interterminology equivalence on all occasions. For example, the requirements of a reporting application may be satisfied by a looser definition of equivalence than a clinical order entry application. The `EQUIVALENCECONTEXT` parameter lets each cross map be associated with the context in which its use is appropriate. At runtime, the `EQUIVALENCECONTEXT` can be provided as a parameter to ETS Equivalence Services to selectively use only those cross maps that are associated with that context.
   - The default `EQUIVALENCECONTEXT` is `SYSTEM`. Cross maps that are flagged with this context will be used by IMP and OMP to determine equivalents in concept lists and the master catalog. If a context is not specified in an equivalence query at runtime, this context is assumed by default.

- If the same cross map is deemed suitable for multiple contexts, it may be repeated several times in the cross maps file, each time with a different `EQUIVALENCECONTEXT`.

3. Load the Map Set files (Section 4.5.3).

4.5.7.3 Invoking Concept Equivalence Services

Reference

Oracle Javadoc for HTB

Table 4–55 lists the principal Concept Equivalence services and methods:

Table 4–55 Service and Methods: Concept Equivalence

Level	Detail
`Package`	`oracle.apps.hct.base`
`Class`	`ETSService`
`Methods`	▪ `getEquivalentConceptIDs` ▪ `isEquivalent`
Class	`Concept`
Methods	▪ `getEquivalentConcepts` ▪ `isEquivalent`
Class	`ConceptList`
Methods	▪ `getEquivalentConcepts`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Healthcare ETS Administrator

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

1. Use the `ETSServiceisEquivalent` method to determine if two concepts are equivalent to each other. Use the `Concept.isEquivalent` method to determine if the `Concept` object is equivalent to another concept. In both of these methods, it is possible to control whether intraterminology equivalence is used or both intraterminology and interterminology equivalence are used. If both are used, it is possible to specify if interterminology mappings should be navigated only to a depth of one cross map from the source terminology, or all possible cross maps should be spanned transitively. Choose the former if the accuracy of equivalence is critical to the application. As more cross maps are navigated away from the source concept the fidelity of meaning should be expected to decrease.
2. Use the `ETSService.getEquivalentConceptIDs` method to get concepts that are equivalent to a concept argument. Use the Concept.getEquivalentConcepts method to get concepts that are equivalent to the `Concept` object. In both of these methods, it is possible to control the type of equivalence used and the depth of cross map traversal if both intraterminology and interterminology equivalence are used. Additionally, it is possible to constrain the returned equivalent concepts to those belonging to a specific terminology version.
3. Use the `ConceptList.getEquivalentConcepts` method to get all concepts from this concept list that are equivalent to a concept argument. Type of equivalence used and the depth of cross map traversal can be controlled when using this method.

4.5.8 Other ETS Support Functions

The following sections describe other ETS support functions:

- Section 4.5.8.1, Searching for a Concept in ETS
- Section 4.5.8.2, Updating Published Coding Scheme Versions
- Section 4.5.8.3, Running the ETS Maintenance Program

4.5.8.1 Searching for a Concept in ETS

Navigation

Table 4–56 summarizes the navigation paths used by this section:

Table 4–56 Navigation Paths: Searching for a Concept in ETS

Function or Window	Navigation Path
Searching for a concept in ETS	`Terminologies > View Versions > View Concepts > Go`
Searching for a term	`Terminologies > View Versions > View Concepts > Go`
Version (search) Window	`Terminologies > View Versions > View Concepts`

Steps

1. Select a terminology to search:
   - Navigate to the Terminologies window by clicking the Terminologies tab.
   - Select a terminology to search, and click the selected terminology's View Versions icon.
2. Select a terminology version to search:
   - In the Versions window, select a version to be searched and click its View Concepts icon.
3. Perform the search:
   - In the Concepts window, select a search type and enter a search value.
   - If you know the **concept code** of the concept, select Concept Code in the Search Type box and enter the code in the Search Value box; the search only returns an exact match.
   - If you are searching for a term, select Descriptions in the Search Type box and enter text in the Search Value box. All concepts with any description (not just the preferred description) that contain a specified pattern are returned.
   - Submit the search.
   - The search results are returned in the Search Results table.
4. Repeat these steps for each coding scheme and version in which the concept may be found.

Guidelines

Cross-terminology, cross-version searching is not supported in the ETS window user interface. The concepts displayed for each terminology (by performing a View Concepts, or by performing a search) are limited to 200 concepts.

Guidelines for Finding an ETS Concept:

- All concepts with any description (not just the preferred description) that contain a specified pattern are returned.
- Concept descriptions have been indexed using Oracle Text, Oracle's integrated full-text retrieval technology.
- The search performs pattern matching. Patterns are always case insensitive, and accept a number of boolean operators.
- The search term must contain at least three non-wildcard characters.

  For example, the patterns in Example 4–5 match the description *Wheelchair*:

Example 4–5 Search Terms Matching Wheelchair

```
wheelchair
wheel% and $chair
%wheel%
```

The patterns in *Example 4–6 do not match* the description *Wheelchair*:

Example 4–6 Search Terms Not Matching Wheelchair

```
%wheel% NOT %chair%
wheel chair
```

Searching with an Oracle Text operator or *wildcard* alone will cause an error. For example, an error results when the following patterns are entered as search terms:

```
%
*
```

Oracle Text maintains a list of *stopwords* called a *stoplist*. Stopwords are those for which Oracle Text does not create an index entry. *This* and *was* are typical stopwords. In the search term

Jack was big

the word *was* is ignored, and phrases such as *Jack is big* and *Jack grew big* are both returned.

Searching on stopwords alone does not return any results—such as Example 4–7:

Example 4–7 Searching on Stopwords

```
this%
this was
```

In addition, the following search term returns an error:

```
this %
```

The word *this* is a stopword, so it is ignored, leaving only the symbol %, which is an invalid search term because it is a wildcard.

> **See Also:** *Oracle Text Application Developer's Guide, Text Reference* (within the Oracle9*i* Database documentation set), for more information about Oracle Text searching, Oracle Text operators and wildcards, and the Oracle Text default stoplist.

4.5.8.2 Updating Published Coding Scheme Versions

After a **coding scheme version** is imported (published), you can update its properties (description, status, and default status) through the ETS window interface. You can use this process, for example, to activate a coding scheme version (Section 4.5.2), or to define it as the default version for messaging (Sections 4.12, 4.13).

Navigation

Table 4–57 summarizes the navigation paths used by this section:

Table 4–57 Navigation Paths: Updating Coding Scheme Versions

Function or Window	Navigation Path
Terminologies Window	`Terminologies`
Update a published version	`Terminologies > View Versions > Update > Apply`
Update Published Version window	`Terminologies > View Versions > Update Published Version`
Versions Window	`Terminologies > View Versions`

Steps

1. Navigate to the Terminologies window.
2. In the Coding Schemes table, locate the target terminology version and click its View Viersions icon.
3. In the Published Versions section of the Versions window, locate the target published version and click its Update icon.
4. In the Update Published Versions window, you can set or modify the following:
   - Description
   - Default Status (Yes / No); selecting Yes removes default status from the selected version
   - Status (Active / Retired)
5. Click Apply; the browser returns to the Versions window and confirms completion of the update.

4.5.8.3 Running the ETS Maintenance Program

The maintenance program performs several database tasks, including cleanup of orphan data, building and synchronizing interMedia indexes, and gathering statistics for the Cost-based Optimizer. In general, it is a good idea to run the maintenance program periodically to keep ETS running optimally.

> **Note:** You should run the ETS Maintenance program each time you apply a patch to ETS.

Steps

To schedule the maintenance program to run:

1. Open a session using the Healthcare ETS Administrator responsibility.
2. Under Concurrent Manager, click Schedule Requests. This launches a process for scheduling a job.
3. Use the list of values in the Name window to select the program called *Healthcare ETS Maintenance Program.*
4. Enter a name for the request in the Name field.

5. Click Next.
6. Complete the scheduling by performing tasks in the Parameters, Schedule, Notifications, and Printing windows.
7. Review your selections in the Summary window and click Submit.
8. Note the Request ID in the Information window and click OK.

See Also:

- *Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests*
- Appendix M, Running Concurrent Programs

4.6 Implementing Audit Services

HTB Auditing Services lets you log and monitor all HTB activities, to monitor security policy and regulation compliance—by recording actions taken by users during sessions. Such actions could include invoking an API, performing a custom function, or other defined events.

HTB Configuration Manager, a GUI tool, lets security administrators define auditing policies. Implementation of HTB Audit Services includes the following steps:

- Enabling HTB Audit Services
- Initializing existing audit event types
- Creating new audit event types
- Invoking HTB Audit Services

Prerequisites

- Implementing Profile Option Services
- Implementing Enterprise Terminology Services

See Also:

- Section 4.4, Implementing Profile Option Services
- Section 4.5, Implementing Enterprise Terminology Services

Login

Log in to Oracle Applications as an administrative user.

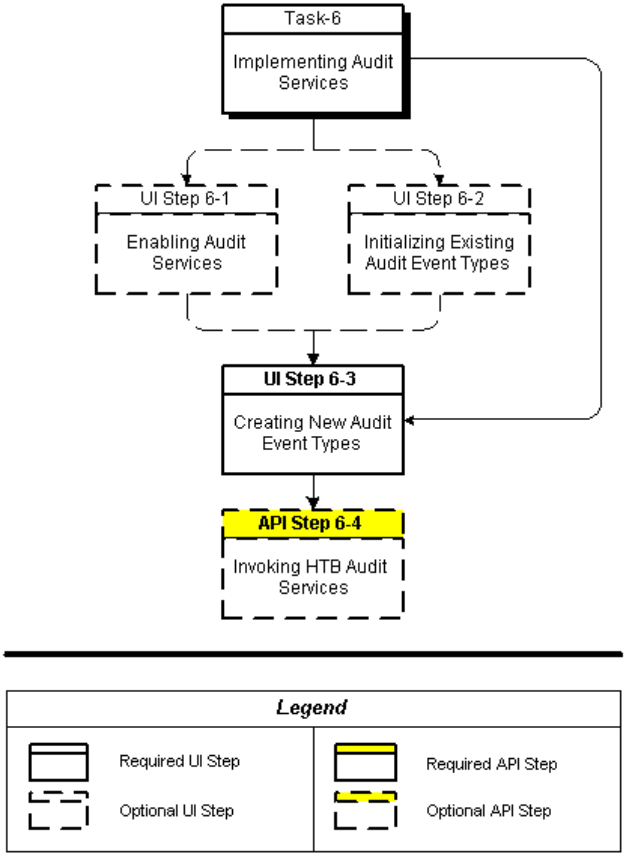
Responsibility

Healthcare Configuration Administrator

Procedures

Figure 4–17 provides an overview of the implementation process for Audit Services:

Figure 4–17 Implementation Process: Audit Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–7, HTB Implementation Procedures: Audit Services

The following sections describe the implementation procedures for Audit Services (referenced by Figure 4–13):

- Section 4.6.1, Enabling Audit Services
- Section 4.6.2, Initializing Existing Audit Event Types
- Section 4.6.3, Creating New Audit Event Types
- Section 4.6.4, Invoking HTB Audit Services
- Section 4.6.5, Attribute Values in Audit Events

4.6.1 Enabling Audit Services

HTB Audit Services can be enabled (turned on) or disabled (turned off) globally. When enabled, audit events of all seeded and user-defined audit event types can be audited. When disabled, Audit Services is not operative.

Login

Log in to Oracle Applications (Self Service Applications) using the user name you defined in Section 4.1.3.

Responsibility

Healthcare Configuration Administrator

Navigation

Table 4–58 summarizes the navigation paths used by this section:

Table 4–58 Navigation Paths: Enabling Audit Services

Function or Window	Navigation Path
Configuring existing Audit Event types	`System Profile Option Values > View Profile Option Values > Update`
Profile Options Values Window	`System Profile Option Values > View Profile Option Values`
Profile Options Window	`System Profile Option Values`
Updating Audit Event Types	`System Profile Option Values > View Profile Option Values > Update`

Steps

1. Navigate to the Profile Options window and select the profile option named `CTB: Auditing ON.`
2. Navigate to the View Profile Option Values window by clicking the glasses icon.
3. Navigate to the Update Profile Option Value window by clicking the pencil icon and select one of the values [Y, N] from the drop-down list.

Default

The `CTB: Auditing ON` profile option defaults to `Y`.

4.6.2 Initializing Existing Audit Event Types

Audit event types can selectively be turned on or off. When both the global auditing flag and a particular audit event type are turned on, events of this particular type are audited by HTB Audit Service.

Login

Log in to Oracle Applications using the user name you defined in Section 4.1.3.

Responsibility

Healthcare Configuration Administrator

Navigation

Table 4–59 summarizes the navigation paths used by this section:

Table 4–59 Navigation Paths: Initializing Existing Audit Event Types

Function or Window	Navigation Path
Configuring existing Audit Event Types	`System Profile Option Values > View Profile Option Values > Update`
Profile Options Values Window	`System Profile Option Values > View Profile Option Values`
Profile Options Window	`System Profile Option Values`
Updating Audit Event Types	`System Profile Option Values > View Profile Option Values > Update`

Steps

1. Navigate to the Profile Options window and select the audit event type.

2. Navigate to the View Profile Option Values window by clicking the glasses icon.
3. Navigate to the Update Profile Option Value window by clicking the pencil icon and select one of the values [Yes, No] from the drop-down list.

Default

A list of HTB audit event types is seeded for HTB use. By default these event types are turned on.

See Also:

- Table E–1, Seeded Audit Events (Appendix E)
- Table D–1, Profile Options (Appendix D)

4.6.3 Creating New Audit Event Types

Applications developed on the HTB Platform can define business audit event types in addition to the seeded event types listed by Table E–1.

For example, an Admitting application might define an audit event type as *Admit Patient*, and monitor events of this type.

Note: Although HTB provides the mechanism to audit business events, it is your responsibility to implement the appropriate audit calls to log such events.

Login

Log in to Oracle Applications using the user name you defined in Section 4.1.3.

Responsibility

Healthcare Application Developer

Navigation

Table 4–60 summarizes the navigation paths used by this section:

Table 4–60 Navigation Paths: Creating New Audit Event Types

Function or Window	Navigation Path
Create Profile Option	`Profile Options > Create Profile Option`

Steps

1. Navigate to the Profile Options page.
2. Navigate to the Create Profile Option page and create the event type with site-level view and update permissions.

4.6.4 Invoking HTB Audit Services

After defining new audit event types, applications can log audit events of these types by calling the Audit Services interface.

Reference

Oracle Javadoc for HTB

Table 4–61 lists the principal methods that relate to Audit Services:

Table 4–61 Service and Methods: Audit Services

Level	Detail
`Package`	`oracle.apps.ctb.auditing`
`Class`	`AuditService`
`Methods`	`createEventLog`

Prerequisite

Creating New Audit Event Types

See Also: Section 4.6.3, Creating New Audit Event Types

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Any responsibility.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

1. Turn on HTB Audit Services (Section 4.6.1) and the audit event type (Section 4.6.2).
2. In the application code, call the `createEventLog` method with the new event type as the value of the `EventType` attribute.

See Also: *Oracle Javadoc for HTB*

4.6.5 Attribute Values in Audit Events

Every entry in the audit trail has the attributes listed by Table 4–62:

Table 4–62 Attributes in Audit Events

Attribute Category	Attribute Name	Description
Access Point Identification	`AccessPointId`	The identifier of the physical device from which the event is logged.
Access Point Identification	`AccessPointSiteId`	The identifier of the enterprise from which the event is logged.
Access Point Identification	`AccessPointType`	The type of access point from which the event originates (IP address, telephone #,...).
Audit Source Identification	`AuditSourceType`	The type of server that logs the event (web server, interface engine, HTB server...).
Event Identification	`EventID`	Unique identifier of the event (automatically generated).
Event Identification	`EventOutcome`	Success or failure codes of the event.
Event Identification	`EventTimestamp`	Timestamp for when the event is logged (automatically generated).
Event Identification	`EventType`	The registered type of event.
Event Identification	EventDescription	A free-text description of the event.
Object Identification	`ObjectId`	The object identifier.
Object Identification	VersionNumber	The version number of the object.
Object Identification	`ObjectIdType`	The type of the object identifier being logged (MRN, SSN,...).
Object Identification	`ObjectLifecycle`	The object lifecycle stage of the event.
Object Identification	`ObjectType`	The type of HTB objects being queried or updated.

Table 4–62 (Cont.) Attributes in Audit Events

Attribute Category	Attribute Name	Description
Object Identification	DataOwner	The owning organization of the object.
Object Identification	DataSubject	The person described by the object.
User Identification	`ApplicationId`	The identifier of the HTB-based application from which the event is logged.
User Identification	SessionId	The identifier of the user session within which the event occurs (automatically derived from session context).
User Identification	`AuthorId`	The identifier of the person who authored the event, which might be different from the logged in user.
User Identification	`LoginId`	The login ID of the current user (automatically derived from session context).
User Identification	`Purpose`	The purpose to access HTB (treatment, payment...); automatically derived from session context.
User Identification	`responsibilityId`	The current responsibility (automatically derived from session context).
User Identification	SecurityProfileId	The organization context within which the event occurs (automatically derived from session context).
User Identification	`TransactionId`	The identifier of the HTB transaction from which the event is logged.

The audit event attributes listed by Table 4–63 have values defined by concept lists:

Table 4–63 Audit Event Attributes: Values Defined by Concept Lists

Attribute Name	Concept List Name	Seeded Values	Extensible?
`AccessPointType`	`CTB_AU_ACCESS_PT_TYPE`	Machine, IPAddress, Telephone	Yes
`AuditSourceType`	`CTB_AU_AUDIT_SOURCE_TYPE`	End User, Web Server, Application Server, Database Server, Security Server, Network Component, Interface Engine, HTB Server	No
`EventOutcome`	`CTB_AU_EVENT_OUTCOME`	Success, Failure	Yes

Table 4–63 (Cont.) Audit Event Attributes: Values Defined by Concept Lists

Attribute Name	Concept List Name	Seeded Values	Extensible?
ObjectLifecycle	CTB_AU_OBJ_LIFE_CYCLE	Origination, Verification, Translation, Access, Identification, Aggregation, Report, Disclosure, Receipt, Archiving, Destruction	No
Purpose	CTB_SEC_PURP	No seeded value	Yes

For extensible attributes with values from concept lists, new values can be added to their concept lists.

For example, applications could add new values to the Purpose concept list, such as Research or Law Enforcement.

> **See Also:** Section 4.5.6.2, Adding Concepts to a Concept List, for details about adding values to concept lists.

4.7 Implementing Consent Services

Overview

HTB Consent Services supports HIPPA Privacy Rule requirements by providing a mix of services that let you create and maintain (i) patient authorizations, (ii) patient consents, and (iii) patient opt-out requests.

Patient Authorization

HTB Consent Services provides a mechanism to protect **personal health information**, captured in several HTB value objects. Every PHI object is owned by a single **data controller**, which can be an organization. Table 4–64 lists PHI objects and the attributes used to identify their owning organizations:

Table 4–64 PHI Objects: Owning Organizations

PHI Object	Owner Organization Attribute
`Person`	`Domain.OrganizationId`
`Patient`	`EnterpriseOrgId`
`MedicalRecord`	`OrgId`
`Encounter`	`MedicalRecordNumber.MRNOrganizationId`
`ClinicalAct`	`MrnOrgPartyId`
`PatientAppointment`	`MedicalRecordNumber.MRNOrganizationId`

Note: Patient Authorization is available for the Person object only when the Flexible Person Domain Service (Section 4.2.4) is enabled.

HTB users access patient PHI through API calls, including two types of processing: (i) use, and (ii) disclosure. When a user logs in, the session is typically associated with an organization context (Section 4.1.10). When the user accesses a PHI object, the access is categorized as *use* if the user's organization context is contained in the owning organization of the PHI object, or as *disclosure* if it is not.

PHI processing is typically performed for certain business purposes, some of which require patient authorization (such as research and marketing). Other purposes, such as treatment, do not require patient authorization. Laws in different jurisdictions may have different requirements regarding which purposes require

patient authorization. HTB is seeded with an empty concept list (CTB_SEC_PURP) that can be configured to contain a list of purposes recognized by an HTB instance, and with another empty concept list (CTB_PCA_PURP) that can be configured to contain a subset of purposes requiring patient authorization.

HTB provides methods that let patients submit authorization requests, represented by a PatientAuthorization object, with the following attributes:

- PatientId: The patient giving the authorization.
- EffectiveTimeLow: Effective date of the authorization.
- EffectiveTimeHigh: Expiration date of the authoriztion.
- Purposes: The list of purposes for which processing is authorized.
- OrgId / OrgName: The data controller that owns the PHI objects to which access is authorized.
- Confidentialities: PHI objects with these confidentiality codes are excluded from authorization.
- RespKey / RespName: Users assigned to this responsibility are authorized access.
- ToOrgId / ToOrgName: Users with this organization context are authorized access.

These attributes are combined to determine which HTB users are authorized to access particular subsets of PHI objects.

See Also:

Oracle Javadoc for HTB [oracle.apps.ctb.security, PatientAuthorization interface], for more information about the PatientAuthorization value object and its attributes.

At runtime every user session is assigned a context attribute that contains the purpose for the session. HTB-based applications control what purposes can be set for a session. For example, a research type of application should not permit users to set *Treatment* as the session purpose. HTB Consent Services validates every access to PHI objects during a session:

- HTB Security Services is called to control access.

- If the session purpose requires patient authorization, access to a PHI object is granted if Consent Services can find a `PatientAuthorization` object that meets the following conditions:
  - `PatientId` matches the PHI object.
  - `EffectiveTimeLow` is less than or equal to `SYSDATE`.
  - `EffectiveTimeHigh` is greater than or equal to `SYSDATE`.
  - Valid `Purpose` is contained in the session context.
  - `OrgId` matches the owning organization of the PHI object.
  - `Confidentialities` do not overlap those of the PHI object.
  - `RespKey` matches the session responsibility.
  - `ToOrgId` contains the session organization context.

Consent and Patient Opt-Out

HTB provides methods that let patients submit consent forms and opt-out requests. Patient consent requests are reflected by `Consent` objects, containing the following attributes:

- `PatientId`: The patient giving consent.
- `Timestamp`: The time when the consent is effective.
- `StatusCode` and `StatusDesc`: The status of the consent. The value of these attributes is derived from concept list `CTB_PCA_CONSENT_STATUS`.

See Also:

Oracle Javadoc for HTB [`oracle.apps.ctb.security, Consent interface`], for more information about the `Consent` value object and its attributes.

Patient opt-out requests are reflected by `OptOut` objects, containing the following attributes:

- `PatientId`: The patient submitting the opt-out request.
- `EffectiveTimeLow` and `EffectiveTimeHigh`: Effective start and end dates of the opt-out request.

- `TypeCode` and `TypeDesc`: The type of opt-out. The value of these attributes is derived from concept list `CTB_PCA_OPT_TYPE`.

See Also:

Oracle Javadoc for HTB [`oracle.apps.ctb.security, OptOut interface`], for more information about the `OptOut` value object and its attributes.

Note: There is no HTB runtime behavior based on consent and opt-out requests. HTB-based applications can utilize these objects (`PatientAuthorization`, `Consent`, `OptOut`) to implement application-specific behavior. For example, in implementing a facility-wide patient directory, opt-out requests can be used to filter out entries for patients who have requested opt-out.

Reference

Oracle Javadoc for HTB

Table 4–65 summarizes HTB Consent service and methods:

Table 4–65 Service and Methods: HTB Consent Service

Level	Detail
`Package`	`oracle.apps.ctb.security`
`Class`	`ConsentService`
`Methods`	▪ `createAuthorizations` ▪ `createConsents` ▪ `createOptOuts` ▪ `findAuthorizations` ▪ `findOptOuts` ▪ `getConsent` ▪ `updateAuthorization` ▪ `updateConsents` ▪ `updateOptOut`

Prerequisites

- Implementing Security Services
- Implementing Organizations
- Implementing Audit Services

See Also:

- Section 4.1, Implementing Security Services
- Section 4.3, Implementing Organizations
- Section 4.6, Implementing Audit Services

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use any responsibility you defined in Section 4.1.4 to which the API function has been granted.

Navigation

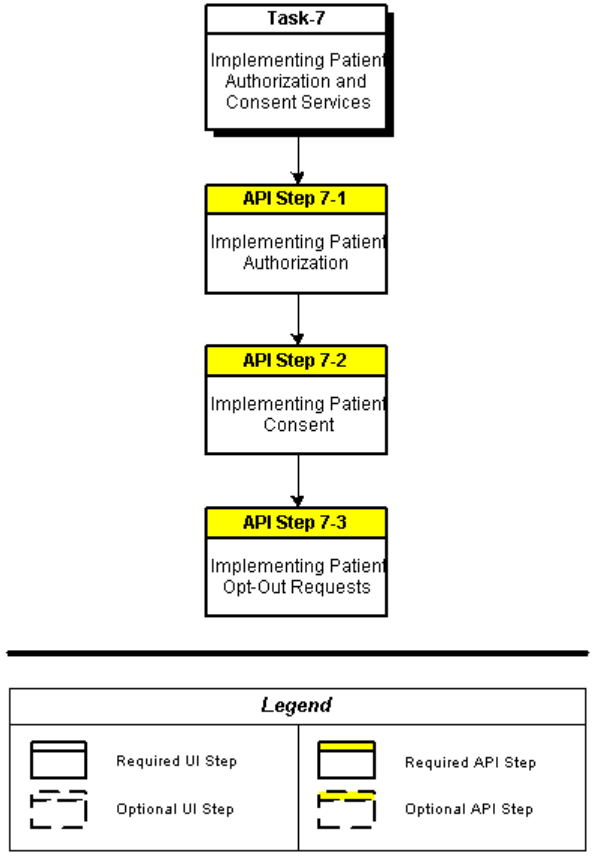
This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Procedures

Figure 4–18 provides an overview of the implementation process for Patient Authorization and Consent:

Figure 4–18 Implementation Process: Patient Authorization and Consent



See Also:

- Figure 3–1, **Implementation Task Process: HTB Platform**
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–8, Implementation Procedures: Patient Authorization and Consent Services

The following sections describe the implementation procedures for Patient Authorization and Consent (referenced by Figure 4–18):

- Section 4.7.1, Implementing Patient Authorization
- Section 4.7.2, Implementing Patient Consent
- Section 4.7.3, Implementing Patient Opt-Out

4.7.1 Implementing Patient Authorization

- Use the `createAuthorizations` method to create new **patient authorization**s—which creates a `PatientAuthorization` value object. A grantee can be (i) an entity that is external to HTB, or (ii) an HTB organization that has a **data controller** role. There cannot be multiple authorizations for the same patient, owning organization, authorizing organization, and responsibility.
- Use the `findAuthorizations` method to query all existing patient authorizations.
- Use the `getAuthorization` method to query a patient authorization value object by its primary key attribute.
- Use the `updateAuthorization` method to update existing patient authorizations.

4.7.2 Implementing Patient Consent

- Use the `createConsents` method to create new patient **consent**s—which creates a `Consent` value object.
- Use the `getConsent` method to query a `Consent` value object by its primary key attribute.
- Use the `updateConsents` method to update existing patient consents.

4.7.3 Implementing Patient Opt-Out

- Use the `createOptOuts` method to create a new patient opt-out—which creates an `OptOut` value object. An out-out gives individuals the opportunity to agree, acquiesce, or object for the following cases:
  - **Facility Directories:** A patient can request that a healthcare facility not publish the individual's name, general condition, religious affiliation, and location in the facility.

  - **For other notifications:** A patient can request that a healthcare organization refrain from disclosing **personal health information** to the individual's family, relatives, friends, or to other persons the individual identifies.
- Use the `findOptOuts` method to query all existing patient opt-outs.
- Use the `getOptOut` method to query an opt-out by its primary key attributes.
- Use the `updateOptOut` method to update patient opt-outs.

4.8 Implementing Administrative Business Services

Administrative Business Services is a set of API services that provide structured access to the HTB data repository for healthcare administration activities. A fundamental goal of HTB is to support the development of applications and the processing of incoming HL7 messages for use in healthcare environments.

Prerequisites

- Implementing Security Services:
  - An administrative security responsibility should be created and assigned all of the protected functions in Organization Services.
  - An administrative account should be created, as a member of the security responsibility.
- Implementing Person Services
- Implementing Organizations
- Implementing Profile Option Services
- Implementing Enterprise Terminology Services
- Implementing Audit Services

See Also:

- Section 4.1, Implementing Security Services
- Section 4.2, Implementing Person Services
- Section 4.3, Implementing Organizations
- Section 4.4, Implementing Profile Option Services
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.6, Implementing Audit Services

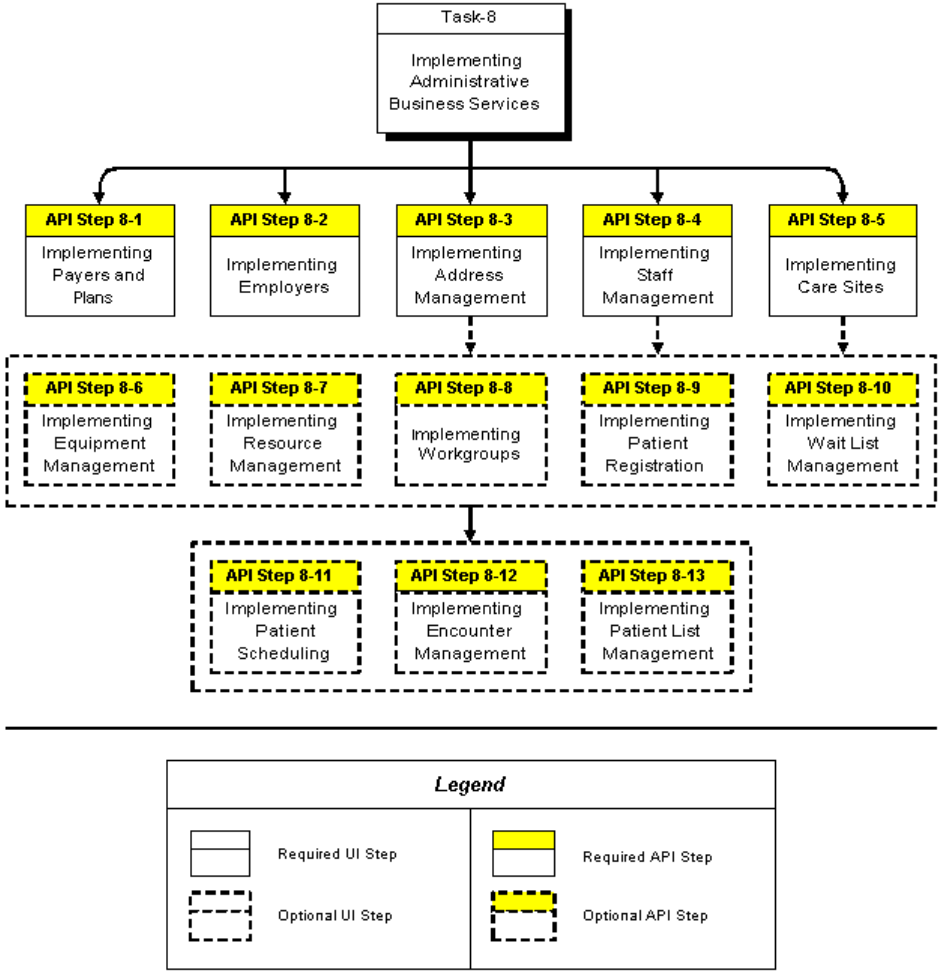
Login

Log in to HTB using the administrative account created in the Prerequisites section (Implementing Security Services).

Procedures

Figure 4–19 provides an overview of the implementation process for Administrative Business Services:

Figure 4–19 Implementation Process: Administrative Business Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–9, HTB Implementation Procedures: Administrative Business Services

The following sections describe the implementation procedures for Administrative Business Services (referenced by Figure 4–19):

- Section 4.8.1, Implementing Payers and Plans
- Section 4.8.2, Implementing Employers
- Section 4.8.3, Implementing Address Management
- Section 4.8.4, Implementing Staff Management
- Section 4.8.5, Implementing Care Sites
- Section 4.8.6, Implementing Equipment Management
- Section 4.8.7, Implementing Resource Management
- Section 4.8.8, Implementing Workgroups
- Section 4.8.9, Implementing Patient Registration
- Section 4.8.10, Implementing Wait List Management
- Section 4.8.11, Implementing Patient Scheduling
- Section 4.8.12, Implementing Encounter Management
- Section 4.8.13, Implementing Patient List Management

4.8.1 Implementing Payers and Plans

Payers and Plans represent insurance carriers and the coverage that they offer. Payer and plan information is referenced during the patient registration process, where an existing payer and plan are associated with the patient registration record, or a new payer and plan are created.

Reference

Oracle Javadoc for HTB

Table 4–66 lists the principal Payer And Plan service and methods:

Table 4–66 Service and Methods: Payers and Plans

Level	Detail
Package	`oracle.apps.ctb.configuration`
Class	`PayerService`
Methods	■ `addCopays` ■ `addPayerContacts` ■ `addPlanContacts` ■ `addPlans` ■ `createPayer` ■ `deleteCopay` ■ `updateCopays` ■ `updatePayer` ■ `updatePayerContacts` ■ `updatePlanContacts` ■ `updatePlans`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in task-1, step 4 (Section 4.1.10).

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

4.8.1.1 Creating Payers

Use the `createPayer` method to create payer organizations. Creating payers involves specifying attributes such as names, dates, and verification status. The following sections describe the business rules governing the creation of payers:

- Names
- Dates
- Verification Status

Names

A payer has a long name and a short name. If there is no short name specified, it defaults to the long name. Payer names must be unique. The payer long name must be unique when compared to other payer long names. The payer short name must also be unique when compared to other payer short names. This business logic ensures that duplicate payers are not created.

Dates

A payer has effective start and end dates. The effective start date defaults to the current date, and the effective end date defaults to *null* (this behavior is typical for most HTB entities).

Verification Status

When creating a payer, you can specify its verification status as Y (Yes) or N (No), indicating if it has been verified. If no value is specified, the verification status defaults to N (No), indicating that this record has not been verified. A payer can be created during initial setup, and as new payers are added by the system administrator. Typically, these records are considered valid and thus verified. When a payer is created during the patient registration process, this data may be less reliable due to variations on names, addresses, and other attributes. The verification status defaults to N (No) during patient registration. The payer record can subsequently be updated to a verification status of Y (Yes), once it has been verified by a system administrator.

4.8.1.2 Updating Payers

Use the `updatePayer` method to update existing payer records. The business rules for payer names, dates, and verification status governing the creation of payers apply to the update of payers as well.

4.8.1.3 Creating Payer Contacts

Use the `addPayerContacts` method to create contact information such as electronic mail address, telephone number, or mailing address associated with a payer. A contact has a contact type attribute, which provides more information

about the use of the contact, such as primary (payer), claims, eligibility. If no contact type is specified, the default contact type is Primary.

Payer, plan, and employer contact information contain identical attributes. The difference is that by using different interfaces, the contact information is associated with either a payer, plan, or employer. Contact types are specific for payers, plans, and employers.

See Also: ■*HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists at bottom of Javadoc page)*

4.8.1.4 Updating Payer Contacts

Use the `updatePayerContacts` method to update existing contacts and addresses. The business rules governing the creation of payer contacts also apply to the update of payer contacts.

4.8.1.5 Creating Plans

Use the `addPlans` method to create plans under an existing payer. Plan creation has similar information and business logic to payer creation, such as long name, short name, effective dates, contact information, and verification status. Note that plans have benefit and copay information that are specific to insurance plans. During the patient registration process, a plan is selected as part of the patient insurance information collected.

4.8.1.6 Updating Plans

Use the `updatePlan` method to update existing plans. The business rules governing the creation of plans also apply to the update of plans.

4.8.1.7 Creating Plan Contacts

Use the `addPlanContacts` method to create new contacts for a plan.

4.8.1.8 Updating Plan Contacts

Use the `updatePlanContacts` method to modify existing contacts for a plan.

4.8.1.9 Creating Copays

Use the `addCopays` method to create copay information under a plan. Copays indicate the copay type and amount for which a patient is responsible. The copay amount must be greater than or equal to zero. A plan can only have one of each

copay type. For example, a plan can only have one copay type and corresponding copay amount per office visit. The exception to this rule is if the copay type is *Other*.

4.8.1.10 Updating Copays

Use the `updateCopay` method to update existing copay information. The business rules governing the creation of copays also apply to updating copays.

4.8.1.11 Deleting Copays

Use the `deleteCopay` method to delete an existing copay from a plan. This method changes the specified copay status to Inactive to indicate that it has been deleted. A deleted copay cannot be reactivated.

4.8.2 Implementing Employers

Employers are associated with patient records during the patient registration process, and are handled similarly to payers in HTB—they both have similar attributes, contact information, and business rules regarding name uniqueness, dates, and verification status.

Reference

Oracle Javadoc for HTB

Table 4–67 lists the principal Employer service and methods:

Table 4–67 Service and Methods: Employers

Level	Detail
Package	`oracle.apps.ctb.configuration`
Class	`EmployerService`
Methods:	▪ `addContacts` ▪ `createEmployer` ▪ `updateContacts` ▪ `updateEmployer`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in task-1, step 4 (Section 4.1.10).

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

4.8.2.1 Creating Employers

Use the `createEmployer` method to create new employers. Employer creation is similar to payer creation. Business rules for payers and plans similarly apply to creating employers.

> **See Also:** Section 4.8.1, Implementing Payers and Plans, for information about business rules regarding name uniqueness, effective dates, and verification status.

4.8.2.2 Updating Employers

Use the `updateEmployer` method to update existing employers. The business rules governing the creation of employers also apply to the update of employers.

4.8.2.3 Creating Employer Contacts

Use the `addContacts` method to add contact information for an employer. Creating employer contacts is similar to payer creation. Business rules for payer and plan contacts similarly apply to creating employer contacts. Contact types are specific to payers, plans, and employers.

> **See Also:** ■*HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page)*

4.8.2.4 Updating Employer Contacts

Use the `updateContacts` method to update existing contact information for an employer. The business rules governing the creation of employer contacts also apply to the update of employer contacts.

4.8.3 Implementing Address Management

Address Management provides a mechanism to centrally manage internal and external addresses. Internal addresses are the locations of internal organizations,

such as those in an internal organization hierarchy, care sites, and equipment, and are stored in the Oracle Human Resource Management model (called the HRMS Model). External addresses are the locations for persons, patients, patient contacts, patient guarantors, clinicians and staff members, and are stored in the Oracle Trading Community Architecture model (called the TCA Model).

Address Management includes the following principal features:

- Single, common interface for internal or external addresses.
- Generic international address style.
- Standard modeling of locations, addresses, and location uses.

There are three fundamental concepts in Address Management:

- **Address**: A physical location.
- **Location**: Associates an address with a person, patient, patient contact,patient guarantor, clinician, staff member or organization; contains additional qualifiers such as mail stop and location use.
- **Location Use**: Qualifies a location for a person or organization (examples: Mail To, Ship To).

> **Note:** Location and location use are created within HTB services that associate an address with an organization, person, patient contact, staff member or clinician; they are not created within Address Management.

Reference

Oracle Javadoc for HTB

Table 4–68 lists the principal Address service and methods:

Table 4–68 Service and Methods: Address

Level	Detail
Package	`oracle.apps.ctb.admin.address`
Class	`AddressService`
Methods	▪ `createAddress` ▪ `updateAddress`

Prerequisites

- Implementing Profile Option Services: A default Business Group must be set.

See Also:

- Section 4.3, Implementing Organizations
- Section 4.4, Implementing Profile Option Services

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

4.8.3.1 Creating Addresses

Use the `createAddress` method to create a new internal or external address record, including the following address characteristics:

- Internal Address
  - Used by internal organization, care site and equipment.
  - Requires a business group set in the `CTB_OS_BUSINESS_GROUP_ID` profile option; the internal address defaults to the business group (set when implementing Organization Management).
  - Uses the generic address style from Oracle HRMS.
- External Address
  - Applied to a person, staff member, clinician, patient contact and patient guarantor.
  - Defined in Oracle Trading Community Architecture (TCA).

When creating a new address, you must indicate if it is internal or external, and you must at least specify Address Line 1, Country Code, and Short Description.

See Also: *Oracle Javadoc for HTB*, for more information about address attributes (`oracle.apps.ctb.admin.address` package; `address` value object)

4.8.3.2 Updating Addresses

Use the `updateAddress` method to update existing address records.

4.8.4 Implementing Staff Management

HTB Staff Management Services lets you create and manage information about staff members and their roles within the healthcare **enterprise**, including credentialing and privileging details. These services are utilized extensively throughout HTB to support both clinical and administrative processes.

HTB Staff Management Services includes the following principal components:

- **Core Staff Management**
  - Staff Member Records
  - Clinician Records
  - Staff Positions
  - Staff Assignment Records
- **Credentials Management**
  - Credential Definitions
  - Credential Records
  - Credential Verification Records
- **Privilege Management**
  - Privilege Definitions
  - Privilege Records

Reference

Oracle Javadoc for HTB

Prerequisites

- **Implementing Person Services: A person record must exist.**

- **Implementing Organizations: An enterprise and an optional facility must exist.**

  See Also:

  - Section 4.2, **Implementing Person Services**
  - Section 4.3, **Implementing Organizations**

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.10.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Procedures

The following sections describe the implementation of Staff Management:

- Section 4.8.4.1, Core Staff Management
- Section 4.8.4.2, Credential Management
- Section 4.8.4.3, Privilege Management

4.8.4.1 Core Staff Management

Core Staff Management lets you create and maintain staff member records—for individuals who may or may not be employees of the enterprise—and manage their assignment to predefined staff positions. The definition and maintenance of these staff positions is supported as well, including the ability to represent **clinician**s in the community who have no contractual relationship with the enterprise. This is particularly useful within the context of PCP tracking, referrals management, and outreach to the broader community of healthcare professionals.

Reference

Oracle Javadoc for HTB

Table 4–69 lists the Core Staff Management services and methods:

Table 4–69 Service and Methods: Core Staff Management

Level	Detail
Package	`oracle.apps.ctb.admin.staffmgmt`
Class	`StaffService`
Methods	▪ `createClinicians` ▪ `createStaffAssignments` ▪ `createStaffMembers` ▪ `createStaffPositions` ▪ `findClinicians` ▪ `findStaffAssignments` ▪ `findStaffMembers` ▪ `findStaffPositions` ▪ `updateClinicians` ▪ `updateStaffAssignments` ▪ `updateStaffMembers` ▪ `updateStaffPositions`

Staff Member and Clinician Records

HTB lets you create and maintain staff member and clinician records associated with a unique person record. Multiple staff member records—but only one clinician record—can be associated with a single person record, but neither is necessary for the creation of the other. The ability to create and maintain such records is a foundational service that supports other functions throughout HTB. For example, staff members must be created as a resource before they can be assigned to either a wait list or an appointment.

- Use the `createStaffMembers` method to create new staff member records. You must specify the person, who must exist in Person Services (Section 4.2), and an organization unit, which must exist in Organization Services (Section 4.3). This process can be initiated in the context of creating a staff assignment, but is typically performed independently.
- Use the `updateStaffMembers` method to update existing staff member records, to reflect changes in status (example: a previously active staff member terminates a contractual relationship with the enterprise) or other staff member

record attributes such as effective dates. Business rules governing the creation of staff members also apply to the updating of staff member records.

- Use the `createClinicians` method to create new clinician records. You must specify the person, who must exist in Person Services.
- Use the `updateClinicians` method to update existing clinician records, to reflect changes in status (example: a clinician retires) or other clinical record attributes such as specialty.

Staff Positions

Staff positions represent enterprise roles defined as relevant to the delivery of care. Staff positions are independent from staff members. You can use the staff position status indicator to activate, suspend, or permanently retire a position. Position records also let you store a job classification, which can represent a standard industry schema (such as HIPAA Provider Taxonomy) or one that is unique to the enterprise. HTB supports the implementation of job classification hierarchies with an unlimited number of tiers, using HTB Enterprise Terminology Services (ETS) (Section 4.5). The organization unit associated with the staff position defines where the position is valid within the enterprise. If the defined unit type for the staff position is *Enterprise*, the position is available for use across the institution. Conversely, if the defined organization unit type is *Facility*, the staff position's use is restricted to the specific facility.

- Use the `createStaffPositions` method to create new staff position records. Note that the defined organization unit (`orgUnit`) for the staff position limits associated staff assignments to those with an organization unit that is the same or a child of the defined unit.
- Use the `updateStaffPositions` method to reflect changes in status (example: a previously valid staff position is retired from use) or other staff position attributes such as effective dates.

Staff Assignment Records

A staff assignment record links staff member records to staff positions. Multiple, active staff assignments associated with a single staff member record are permissible as are multiple assignments that associate the same staff member record and staff position. This lets you assign staff members to multiple departments within an enterprise, which is common practice within the Healthcare industry.

Staff assignments are also scoped through the association of an organization unit. This lets the enterprise define an organizational scope for a staff assignment that can be more restrictive than that of the associated staff position.

- Use the `createStaffAssignments` method to link staff member records to staff positions—assigning a staff member to a staff position. If the staff member record or staff position or both do not already exist, their creation is initiated in the process of creating a staff assignment.
- Use the `updateStaffAssignments` method to update staff member assignments, to reflect changes in status (example: an active staff assignment is temporarily suspended from use) or other staff assignment attributes such as effective dates. Business rules governing the creation of staff assignments also apply to the updating of staff assignments.

4.8.4.2 Credential Management

Credential Management provides a framework for documenting clinician **credentials**, including related verification activities.

Reference

Oracle Javadoc for HTB

Table 4–70 lists the Staff Management services and methods for credentials:

Table 4–70 Service and Methods: Staff Management Credentials

Level	Detail
Package	`oracle.apps.ctb.admin.staffmgmt.credential`
Class	`CredentialService`
Methods	▪ `createCredentialDefinitions` ▪ `createCredentials` ▪ `createExternalOrgUnits` ▪ `createVerifications` ▪ `findCredentialDefinitions` ▪ `findCredentials` ▪ `findExternalOrgUnits` ▪ `findVerifications` ▪ `updateCredentialDefinitions` ▪ `updateCredentials` ▪ `updateExternalOrgUnits` ▪ `updateVerifications`

Credential Categories

HTB includes the following predefined categories for documenting credentials:

- Board Certificates
- Certifications
- Education
- Liability Insurance
- Licenses
- Malpractice Claims
- Previous Employment
- References
- Registrations
- Training

Credential Management also includes seeded lists of subcategories for many of these categories. Some of the predefined credential categories (such as education) represent areas of overlap between credentialing and traditional human resources functionality. In such cases, these credential records can also be accessed through Oracle HRMS. The enterprise can augment these predefined categories in order to meet its unique organizational requirements.

See Also:

- Seeded concept list `CTB_SCP_CRED_DEF_CODE`, for a list of seeded credential categories.
- *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page),* for seeded concept lists.

- Use the `createCredentialDefinitions` method to define new credential categories.
- Use the `updateCredentialDefinitions` method to update or modify credential categories.

Credential Records

The credential record is used to document the details of an individual clinician's credentials, including the issuing organization, completion date and other related

information. A credential record is associated with only one clinician record, while a clinician record can have many associated credential records.

- Use the `createCredentials` method to create records of individual clinician credentials.
- Use the `updateCredentials` method to update or modify existing credential records, including those requiring verification.

Credential Verification Record

HTB Credentials Management lets you collect and maintain data related to the **verification** of credentials—including both successful and unsuccessful verification attempts. Verification records are scoped by organization units to support primary source verification requirements. Some verified credentials remain valid indefinitely, while others may be subject to expiration or revocation—and consequently require ongoing monitoring as well as reverification on a cyclical basis. A verification record is associated with only one credential record, while a credential record can have many associated verification records.

- Use the `createVerifications` method to create records of credential verification activities.
- Use the `UpdateVerifications` method to update verification records, to reflect changes in status (example: a verification that was previously pending is now completed) or other verification record attributes.

4.8.4.3 Privilege Management

Privilege Management Services lets enterprises define and limit the scope of clinical services that individual licensed practitioners are permitted to perform within the institution. The extend of the services supported by the enterprise is defined through creation and maintenance of privilege definitions. These definitions are the basis for formal clinician requests to deliver clinical services utilizing the enterprises's resources (such as the use of the operating room to perform surgery). A clinician's request and its subsequent disposition (granted or denied) are documented in HTB as a privilege.

Reference

Oracle Javadoc for HTB

Table 4–71 lists the Staff Management services and methods for privileges:

Table 4–71 Service and Methods: Staff Management Privileges

Level	Detail
Package	`oracle.apps.ctb.admin.staffmgmt.privilege`
Class	`PrivilegeService`
Methods	■ `createPrivilegeDefinitions` ■ `createPrivileges` ■ `findPrivilegeDefinitions` ■ `findPrivileges` ■ `updatePrivilegeDefinitions` ■ `updatePrivileges`

Privilege Definition

HTB lets you create privilege definitions tailored to your enterprise's unique business requirements. This functionality includes supporting the optional association of privilege definitions with procedure code terminologies maintained in Enterprise Terminology Services (ETS). The scoping of privilege definitions by organization units is also supported. This association is useful for the development of privilege catalogs that correspond to enterprise organization units. Additionally, privilege definitions can be grouped through association with a staff position. This functionality provides the enterprise with a mechanism for defining sets of privileges that are routinely requested and granted together. These groupings of privilege definitions by staff positions may correspond to specialties, sub-specialties or other internally meaningful subdivisions.

- Use the `createPrivilegeDefinitions` method to define standardized privilege definitions that represent an institutions' unique business practices.
- Use the `updatePrivilegeDefinitions` method to update or modify existing privilege definitions to reflect periodic policy changes.

Privilege Life Cycle

HTB Privilege Management lets enterprises track privileges from request to grant through ongoing maintenance. A privilege exists at the intersection of a staff assignment and a privilege definition. A single staff assignment can have one or more privileges associated with a specific privilege definition. The privilege record functions as a repository for any information relevant to the staff member's exercising of the specified privilege definition. Privileges let the enterprise enforce a customized classification schema (example: emergency versus full privileges),

indicate any supervision requirements, specify the status of the privilege (example: new versus active), and define the time period for which the privilege is valid.

- Use the `createPrivileges` method to create privilege records associated with a staff assignment. This method is used primarily to document a request for a particular privilege, but can also be used for the granting of emergency privileges. The disposition of privilege requests is recorded through the `updatePrivileges` method.
- Use the `updatePrivileges` method to record the disposition of privilege requests entered through the `createPrivileges` method, and for the ongoing management of those privilege records, including suspensions or other changes in status.

4.8.5 Implementing Care Sites

Care Site Management supports the management of **care site**s within a healthcare facility. It provides functions to create and maintain care sites, and manage their day-to-day operations. The hierarchy of care site management is: *Facility* > *Room* > *Bed*, with facility at the highest level of the hierarchy. A care site is a defined location—typically a room or bed registered as a resource, for scheduling purposes. You can independently define a care site, associate it with a facility location, and optionally assign it to a **practice setting**.

Each practice setting has a list of care sites (within the facility they are part of) that are allocated to patients as they arrive. One care site can be used by more than one practice setting and more than one patient can be assigned to a single care site. A waiting room is one example. Beds can also be *occupied* by multiple patients. For example, a patient may be in the process of being discharged while another patient has been assigned to the bed, but has not yet arrived (pending arrival).

Care Site Management and Encounter Management are closely related. Care Site Management includes setup and maintenance of rooms and beds, while Encounter Management generates activity involving care sites, such as assigning patients to rooms and beds.

Reference

Oracle Javadoc for HTB

Table 4–72 lists the principal Care Site service and methods:

Table 4–72 Service and Methods: Care Sites

Level	Detail
Package	`oracle.apps.ctb.admin.caresitemgmt`
Class	`CareSiteMgmtService`
Methods	■ `createCareSites` ■ `findCareSites` ■ `updateCareSites`

Prerequisites

- Implementing Organizations:
  - An active facility organization must exist.
  - An active practice setting must exist.
- Implementing Address Management: An internal address must be defined.

See Also:

- Section 4.3, Implementing Organizations
- Section 4.8.3, Implementing Address Management

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal care site tasks:

- Section 4.8.5.1, Creating Care Sites
- Section 4.8.5.2, Updating Care Sites

4.8.5.1 Creating Care Sites

Use the `createCareSites` method to create new care sites. Care sites have attributes common to all care sites, including:

- Name
- Effective Time Low
- Owning Organization Identifier
- Location Identifier

A care site can be one of two types: Room or bed. Each type has characteristics specific to that type. For example, a room not only has care site attributes, but it must also have *Room Number Text* and *Shared Indicator* attributes. A bed must have the same owning facility as the room to which it is assigned and the *Bed Number Text* attribute.

In addition to the care site type, the custom type, which is user extensible, lets you customize the type of room or bed. For example, you can identify a room and bed respectively as:

  - Exam room
  - Waiting room
  - Inpatient bed
  - Outpatient bed

Business rules governing the creation of care sites follow:

- The owning organization must have an organizational unit defined with a Facility role.
- The location or address of each facility housing a care site must be defined and have the same address as the owning organization.
- If you set the care site shared indicator flag to *Y* (Yes), you can assign the care site to more than one practice setting (this applies only to rooms.)

Care site subcomponents include:

- **Activity**: Requesting a date and time for specific activities, such as housekeeping or maintenance repairs.
- **Assignment**: Assigning one care site to another or assigning a care site to a practice setting. For example, assigning a bed to a room or a room to a practice

setting. If the room care site shared indicator is set to *Y* (Yes), you can assign multiple rooms to practice settings.

When defining an assignment, you must confirm that both care sites are owned by the same facility.

- **Contact**: The name and phone number of whom to contact about the care site.
- **Feature**: The available features for each care site. Examples include oxygen, room darkening and television.
- **Inactive Period**: The start and end dates and times of temporary closure for a particular care site.
- **Specialty**: Areas of expertise (such as cardiology or pediatrics) associated with a care site. Specialties are defined in Staff Management and referenced in Care Site Management, Encounter Management, Wait List Management, and Organization Management (Practice Settings).

4.8.5.2 Updating Care Sites

Use the `updateCareSites` method to update existing care sites. Because a care site has different attributes that are determined at the time of creation by its type, you cannot update the care site type.

You can only update a room's shared indicator from *Y* (Yes) to *N* (No) if a room is assigned to only one practice setting.

4.8.6 Implementing Equipment Management

Equipment Management facilitates the creation and maintenance of equipment within healthcare enterprises. Similar to care site treatment, equipment is registered as a resource for scheduling purposes.

Reference

Oracle Javadoc for HTB

Table 4–73 lists the principal Equipment Management service and methods:

Table 4–73 Service and Methods: Equipment

Level	Detail
Package	`oracle.apps.ctb.admin.equipment`
Class	`EquipmentService`

Table 4–73 (Cont.) Service and Methods: Equipment

Level	Detail
Methods	■ `createEquipment` ■ `updateEquipment`

Prerequisites

- Implementing Organizations: An active organization must exist.
- Implementing Address Management: An internal address must be defined.

See Also:

- Section 4.3, Implementing Organizations
- Section 4.8.3, Implementing Address Management

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal tasks relating to equipment management:

- Section 4.8.6.1, Creating Equipment
- Section 4.8.6.2, Updating Equipment

4.8.6.1 Creating Equipment

Use the `createEquipment` method to create equipment. Business rules governing the creation of equipment follow:

- You can define an organizational unit with any role as the owning organization.
- You must define the internal address where the equipment is located.

- If you set the equipment shared indicator flag to *Y* (Yes), you can assign the equipment to more than one organization or care site.

Equipment Management components include:

- **Activity**: Requesting a date and time for specific activities, such as maintenance repairs.
- **Assignment**: Assigning equipment to a specific care site or organization. If the equipment shared indicator is set to *Y* (Yes), you can assign it to multiple care sites and organizations.

  When defining an assignment, you must confirm that the *assigned to organization* defined is a child of the *equipment owning organization*.
- **Contact**: The contact name and telephone number related to particular equipment.
- **Feature**: The features offered for each piece of equipment; *portable*, for example. This attribute is user extensible.
- **Inactive Period**: The start and end dates and times of temporary down time for given equipment.

4.8.6.2 Updating Equipment

Use the `updateEquipment` method to update a collection of existing equipment.

You can only update the equipment shared indicator from *Y* (Yes) to *N* (No) if the equipment is assigned to only one practice setting.

4.8.7 Implementing Resource Management

Resource Management provides methods to create, update and find resources, such as **care site**s (rooms, beds), equipment, and staff members. The Resource Service provides a central repository for registering, grouping, and searching different types of resources for scheduling purposes. You can identify different entities as resources so they can be flagged as *schedulable* or *non-schedulable*. A scheduling application can select *schedulable* resources for appointments.

A Resource Group is a collection of resources. Groups can also be contained within other groups. You can use the grouping of similar resources to indicate that the members of the group can perform similar tasks or have similar features or qualifications, or you can use the grouping of dissimilar resources to define a set of resources required to perform a particular procedure.

Reference

Oracle Javadoc for HTB

Table 4–74 lists the principal Resource Management service and methods:

Table 4–74 Service and Methods: Resource Management

Level	Detail
Package	`oracle.apps.ctb.admin.resource`
Class	`ResourceService`
Methods	▪ `createResources` ▪ `updateResources`

Prerequisites

- Implementing Organizations: An active organization location must exist.
- Implementing Staff Management: Staff members must be defined.
- Implementing Care Sites: A care site must be defined.
- Implementing Equipment Management: Equipment must be defined.

See Also:

- Section 4.3, Implementing Organizations
- Section 4.8.4, Implementing Staff Management
- Section 4.8.5, Implementing Care Sites
- Section 4.8.6, Implementing Equipment Management

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal resource management tasks:

- Section 4.8.7.1, Creating Resources
- Section 4.8.7.2, Creating Resource Groups
- Section 4.8.7.3, Updating Resources

4.8.7.1 Creating Resources

Use the `createResources` method to create resources. There are four resource types available for this purpose:

- **Care Site Resource:** Registers an existing care site as a resource.
- **Equipment Resource:** Registers existing equipment as a resource.
- **Staff Member Resource:** Identifies enterprise staff member (a **practitioner**) who are registered as a resource.
- **Group:** Defines a collection of similar or dissimilar resources, including other groups.

The *schedulable* indicator is common to all four types of resources. It indicates to a scheduling application if a resource can be reserved for appointments. For groups, if you set this indicator flag to *Y* (Yes), its members must also have their *schedulable* indicator set to *Y* (Yes).

4.8.7.2 Creating Resource Groups

Use the `createResources` method to create resource groups. Resource groups define collections of individual resources or other resource groups. Resource groups are used to logically group resources based on common or disparate attributes. A sample group may consist of the following resources:

- Exam room
- X-Ray machine
- Radiologist

Group attributes include the following:

- **Name:** The name of each resource group must be unique within the owning organization.

- **Owning Organization:** An organization unit with any role; all members within a group must belong to the same organization or one of its child organizations within the enterprise hierarchy.
- **Members:** Individual resources or resource groups, with a mix of similar or dissimilar resource types. The resource or resource group members must belong to the same owning organization or belong to a child of the owning organization in the enterprise hierarchy.
- **Type:** User-extensible type of resource group. The seeded type Medical Service defines a group of staff member resources or care site resources for reporting purposes. Encounters and practice setting organizations can have medical services associated to them. Medical services typically represent departmental affiliations in a facility and are specific to the facility. Examples include:
  - Cardiology
  - Pediatrics

See Also: Section 4.8.12, Implementing Encounter Management

4.8.7.3 Updating Resources

Use the `updateResources` method to update existing resources or resource groups. The *schedulable* indicator defines if a resource can be reserved for appointments. When updating resources, if you set this indicator flag to *Y* (Yes), its members must also have their *schedulable* indicator set to *Y* (Yes).

4.8.8 Implementing Workgroups

Membership in a workgroup provides healthcare organization staff members with common access to certain business functions and data elements within a clinical software system. Staff members can belong to different workgroups, depending upon their staff assignments.

Reference

Oracle Javadoc for HTB

Table 4–75 lists the principal Workgroup service and methods:

Table 4–75 Service and Methods: Workgroups

Level	Detail
Package	`oracle.apps.ctb.admin.staff.workgroups`

Table 4–75 (Cont.) Service and Methods: Workgroups

Level	Detail
Class	`WorkgroupService`
Methods	▪ `addWorkgroup` ▪ `inactivateWorkgroup` ▪ `updateWorkgroup`

Prerequisites

Implementing Staff Management: Staff member and staff assignment records must exist.

See Also: Section 4.8.4, Implementing Staff Management

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.10.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal tasks relating to workgroups:

- Section 4.8.8.1, Creating Workgroups
- Section 4.8.8.2, Updating Workgroups
- Section 4.8.8.3, Inactivating Workgroups

4.8.8.1 Creating Workgroups

Use the `addWorkgroup` method to create new workgroups. This method creates a workgroup within an organization to which workgroup members can be added. Business rules governing creation of workgroups are described in the following sections:

- Organizations
- Workgroup Membership
- Workgroup Type
- Dates

Organizations

A workgroup must be associated with an enterprise or facility organization, which lets members of the workgroup access functions or data associated with that organization. A workgroup can only be associated with one organization unit. The workgroup name is mandatory and must be unique within the organization. However, the same workgroup name can be used at another organization, such as the enterprise itself, or another facility within the same enterprise.

Workgroup Membership

Healthcare staff members acquire membership in workgroups through their staff assignments. This membership lets staff members access certain functions and data sets, depending upon the particular assignments they hold. The workgroup and staff assignment can belong to either an enterprise organization or to a facility organization. In either case, the workgroup and staff assignment organizations must exist within the same enterprise or roll up to the same enterprise. If one belongs to an enterprise and the other belongs to a facility, the facility must be subordinate to the enterprise. If both belong to the same or different facilities, the facilities must be subordinate to the same enterprise.

A single staff assignment cannot have overlapping membership dates in the workgroup. Use the `updateWorkgroup` method to change or extend an assignment membership.

Workgroup Type

The workgroup type is mandatory and categorizes the workgroup into a functional classification. The customer is responsible for populating the `CTB_EM_MEDICAL_SERVICE_CODE` concept list with the desired values for workgroup type.

See Also:

- *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page).*
- Section 4.5, Implementing Enterprise Terminology Services, for information about registering specific lookup values.

Dates

The workgroup has a set of effective start and end dates. In addition to the requirement that the start date must precede the end date, these dates must be consistent with the workgroup membership start and end dates and the organization existence time low and high. The workgroup effective start date defaults to the organization existence time low or the current date, whichever is greater. If the organization has a future existence time high, the workgroup effective end date defaults to that date, or can be set to an earlier date.

Similarly, when assigning members to a workgroup, the workgroup membership effective start date defaults to the workgroup effective date or current date, whichever is greater. If the workgroup has an effective end date in the future, the membership effective end date defaults to that date, or can be set to an earlier date.

4.8.8.2 Updating Workgroups

Use the `updateWorkgroup` method to update an existing workgroup. Business rules governing the creation of workgroups also apply to the update of workgroups, although there are some additional business rules regarding retroactive date changes and the update of the effective end date of a workgroup.

Retroactive changes to start and end dates that precede the current date are not allowed. A workgroup cannot be created with its effective start and end dates set to a prior period. If a workgroup is active, its effective start date cannot be updated. If a workgroup has expired, its effective end date cannot be updated.

Use the `updateWorkgroup` method to inactivate a workgroup on a future date.

See Also: Section 4.8.8.3, Inactivating Workgroups

4.8.8.3 Inactivating Workgroups

Use the `inactivateWorkgroup` method to inactivate a workgroup by setting its effective end date. This date must be less than or equal to the organization existence time high. Consistent with restrictions on retroactive changes, the workgroup's effective end date cannot precede the current date. If the workgroup is currently active, the effective end date defaults to the current date. If the workgroup has a future effective start date, the effective end date defaults to the same date.

To inactivate the workgroup on a specific date, use the `updateWorkgroup` method to set the effective end date to a future date.

Note that neither the `inactivateWorkgroup` method nor the `updateWorkgroup` method can inactivate a workgroup containing active memberships. If active

memberships do exist, you must set their end dates before inactivating the workgroup.

4.8.9 Implementing Patient Registration

Patient Registration maintains the foundational data set necessary to support encounter management and patient accounting activities within a healthcare organization. The patient registration process includes the capture of patient demographic, medical, insurance, and employer information.

Patient Registration is integrated within the following HTB services:

- **Person Services:** Provides a single repository for all patient data. Patient data can be masked (using patient pseudonym) for privacy or business reasons, while maintaining a single identifying record for the person.
- **Encounter Management:** Patients participate in and are subjects of encounters.
- **Patient List:** Patient census data is displayed in patient lists.

Reference

Oracle Javadoc for HTB

Table 4–76 lists the principal Patient Registration service and methods:

Table 4–76 Service and Methods: Patient Registration

Level	Detail
Package	`oracle.apps.ctb.admin.patient.registration`
Class	`PatientService`

Table 4–76 (Cont.) Service and Methods: Patient Registration

Level	Detail
Methods	■ assignEmployers ■ assignInsurances ■ assignPcps ■ createMedicalRecords ■ createPatient ■ createPatientContacts ■ createPatientGuarantors ■ findEmployers ■ findInsurances ■ findPatientContacts ■ findPatientGuarantors ■ getMedicalRecordByExternalid ■ getMedicalRecordByMedicalRecordNumber ■ getMedicalRecords ■ getPatient ■ getPcps ■ updateEmployers ■ updateInsurances ■ updateMedicalRecords ■ updatePatient ■ updatePatientContacts ■ updatePatientGuarantors ■ updatePcps

Prerequisites

- **Implementing Organizations:**
  - An enterprise must exist to create a patient record.
  - An enterprise or facility organization must exist—with the role of MRI (Medical Record Issuer), indicating the ability to create medical records.
- **Implementing Person Services: A** person record must exist.

- **Implementing Payers and Plans:** A payer record and a plan record must exist to assign a patient to an existing payer or plan; otherwise, you can create a new payer or plan.
- **Implementing Employers:** An employer record must exist to assign a patient to an existing employer; otherwise, you can create a new employer.
- **Implementing Staff Management:** A staff member record must exist to assign a patient to an existing primary care provider; otherwise, you can specify a non-registered provider.

See Also:

- Section 4.2.5, Implementing Person Management
- Section 4.3, Implementing Organizations
- Section 4.8.1, Implementing Payers and Plans
- Section 4.8.2, Implementing Employers
- Section 4.8.4, Implementing Staff Management

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.10.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Procedures

The following sections describe HTB methods that let you create and maintain patient records, and associate them with existing payers, plans, employers staff members, contacts, and guarantors:

- Section 4.8.9.1, Creating Patients
- Section 4.8.9.2, Updating Patients
- Section 4.8.9.3, Assigning Insurance Information to Patients

- Section 4.8.9.4, Updating Insurance Information Assigned to Patients
- Section 4.8.9.5, Assigning Employer Information to Patients
- Section 4.8.9.6, Updating Employer Information Assigned to Patients
- Section 4.8.9.7, Assigning Primary Care Physician Information to Patients
- Section 4.8.9.8, Updating Primary Care Physician Information Assigned to Patients
- Section 4.8.9.9, Creating Patient Medical Records
- Section 4.8.9.10, Updating Patient Medical Records
- Section 4.8.9.11, Creating Patient Contacts
- Section 4.8.9.12, Updating Patient Contacts
- Section 4.8.9.13, Creating Patient Guarantors
- Section 4.8.9.14, Updating Patient Guarantors

4.8.9.1 Creating Patients

Use the `createPatient` method to create new patients from an existing person record and associate them with an enterprise organization. You can specify the following optional information for the patient:

- **Religious:** Place of worship.
- **Tribal Membership:** Tribal Affiliation (a more specific classification of ethnicity; for example, if ethnicity is Native American, tribal membership might be Cherokee Nation, Oklahoma).
- **Medical:** Primary care physician.
- **Insurance:** Insurance carrier, plan, and dates of coverage.
- **Employment:** Employer and dates of employment.

Use the `createPatient` method to create a temporary patient from an existing temporary person record.

See Also:

- Section 4.2.5, Implementing Person Management

4.8.9.2 Updating Patients

Use the `updatePatient` method to update information for an existing patient. Business rules governing the creation of patients also apply to updating patients.

4.8.9.3 Assigning Insurance Information to Patients

Use the `assignInsurances` method to assign insurance information to a patient. This includes payer, plan, and policy holder employer information to be associated with a patient, as well as claim, copay, and eligibility information. Note that this method creates new employer records if necessary. Insurance claim filing order must be in numerical sequence. The copay amount must be greater than or equal to zero.

4.8.9.4 Updating Insurance Information Assigned to Patients

Use the `updateInsurances` method to update information for an existing patient. Business rules governing the creation of patient insurance information also apply to updating such information.

4.8.9.5 Assigning Employer Information to Patients

Use the `assignEmployers` method to assign employer information to patients, including the policy holder employer, patient dates of employment, and patient job title.

You can assign the patient to an existing policy holder employer by specifying an existing employer identifier. Or you can create a new employer record to assign to the patient, by specifying the employer attributes (such as name, address, phone number). You cannot specify both. If you do, only the employer identifier is accepted, and the other attribute information is ignored.

The employment start date must precede the employment end date. Multiple assignments to the same employer are allowed, as long as the employment start and end dates of each assignment do not overlap. Multiple assignments with overlapping dates are not allowed.

> **See Also:** Section 4.8.2, Implementing Employers, to create a new employer record with unverified status.

4.8.9.6 Updating Employer Information Assigned to Patients

Use the `updateEmployers` method to update employment assignment information for a patient. Do not confuse this method with the (`EmployerService`) `updateEmployer` interface, which is used to update existing

employer records created with the (`EmployerService`) `createEmployer` interface.

The (`PatientService`) `updateEmployers` interface lets you update the patient job title and effective dates of employment only, but not the employer information.

4.8.9.7 Assigning Primary Care Physician Information to Patients

Use the `assignPcps` method to assign a primary care physician (PCP) to a patient. You can specify an existing staff member, but you cannot create a new staff member record during this process (as you can with assigning a payer or employer to the patient). If the PCP does not exist as a staff member, you can specify a non-registered provider name and telephone number as the PCP. These two options are mutually exclusive.

The PCP-to-patient assignment has effective start and end dates, where the effective start date must precede the end date. Multiple assignments of the same PCP to the same patient are permitted, as long as the assignment effective dates do not overlap.

4.8.9.8 Updating Primary Care Physician Information Assigned to Patients

Use the `updatePcps` method to update existing primary care provider information assigned to a patient. Business rules governing the creation of PCP information assigned to patients also apply for updating such information.

4.8.9.9 Creating Patient Medical Records

Use the `createMedicalRecords` method to create a medical record for a patient at an enterprise or facility organization, once a patient receives treatment at the organization (this is usually a facility, such as a hospital or clinic). The organization must have a role of MRI (Medical Record Issuer), indicating the ability to create medical records. This process generates a unique **medical record number** for the patient at that organization. You can optionally flag the medical record as VIP, indicating that the patient should receive special treatment.

4.8.9.10 Updating Patient Medical Records

Use the `updateMedicalRecords` method to update existing medical records for a patient. Business rules governing the creation of patient medical records also apply to updating such records.

4.8.9.11 Creating Patient Contacts

Use the `createPatientContacts` method to create contact records for a patient. The type of contact must be emergency contact or next of kin. The record can include the following information:

- Contact person relationship to the patient
- Contact name
- Contact address
- Contact telephone number
- Contact e-mail address
- Effective dates of the information

A patient can have multiple contact records.

4.8.9.12 Updating Patient Contacts

Use the `updatePatientContacts` method to update existing contact records for a patient. Business rules governing the creation of patient contacts also apply to updating such records.

4.8.9.13 Creating Patient Guarantors

Use the `createPatientPersonGuarantors` method to create guarantor records for a patient. The record can include the following information:

- Guarantor relationship to the patient
- Guarantor name
- Guarantor address
- Guarantor telephone number
- Guarantor e-mail address
- Guarantor employer
- Effective dates of the information

A patient can have multiple guarantor records.

4.8.9.14 Updating Patient Guarantors

Use the `updatePatientPersonGuarantors` method to update existing guarantor records for a patient. Business rules governing the creation of patient guarantors also apply to updating such records.

4.8.10 Implementing Wait List Management

Wait List Management lets you create and manage wait lists and their entries. Organized by resource (staff member, care site, equipment or resource group), appointment type, appointment series, or organization unit, wait lists are composed of entries, representing patients waiting for appointments. Principal features include the following:

- Creation and management of wait lists
- Creation and management of wait list entries
- Linking **clinical act**s to a wait list entry patient record
- Versioning and version queries:

  All wait list and wait list entry tables are versioned. Wait List Management extends Versioning Services to support the following queries:

  - Point-in-Time
  - Version History
  - Next Version
  - Previous Version
  - Audit (created by, created when)

Reference

Oracle Javadoc for HTB

Table 4–77 lists the principal Wait List Management service and methods:

Table 4–77 Service and Methods: Wait List Management

Level	Detail
`Package`	`oracle.apps.ctb.admin.waitlist`
`Class`	`WaitListService`

Table 4–77 (Cont.) Service and Methods: Wait List Management

Level	Detail
Methods	■ createWaitList ■ createWaitListEntry ■ deleteWaitList ■ deleteWaitListEntry ■ findWaitListEntries ■ findWaitLists ■ getWaitList ■ getWaitListEntry ■ moveWaitListEntries ■ updateWaitList ■ updateWaitListEntry

Prerequisites

- Implementing Organizations: An organization unit must exist to be associated with a wait list.
- Implementing Resource Management: A resource must exist to be associated with a wait list. Staff members, care sites, equipment and resource groups must be created as resources before they can be associated with a wait list.
- Implementing Patient Registration: A patient with a medical record must exist for a wait list entry to be created.
- Implementing Patient Scheduling: An appointment type and an appointment series must exist to be associated with a wait list.
- Implementing Enterprise Terminology Services: ETS must be implemented to facilitate creation of concept lists.
- Creating a Concept List: Values for extensible Wait List Management concept lists should be created as required.
- Defining Concept List Values (*Oracle Javadoc HTB Concept Lists Index*):
  - Values for appointment type names must be defined for organization appointment types to be created, and for appointment types to be associated with wait lists. Scheduling and Encounter Services share the concept list Encounter Type Code (the same values are used for encounter types and appointment types).

- Values for Action Reason must exist.
- Values for Specialty Type Code must exist for a speciality to be associated with a wait list. Wait List Services use the same concept list for speciality as Staff Management.

See Also:

- Section 4.3, Implementing Organizations
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.5.6.1, Creating a Concept List
- Section 4.8.7, Implementing Resource Management
- Section 4.8.9, Implementing Patient Registration
- Section 4.8.11, Implementing Patient Scheduling
- Appendix H, Empty Concept Lists
- *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page)*

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal wait list tasks:

- Section 4.8.10.1, Creating Wait Lists
- Section 4.8.10.2, Creating Wait List Entries
- Section 4.8.10.3, Deleting Wait Lists
- Section 4.8.10.4, Deleting Wait List Entries

4.8.10.1 Creating Wait Lists

Use the `createWaitList` method to create new wait lists. You can create wait lists of the following encounter classes:

- Inpatient
- Ambulatory (outpatient)
- Emergency

Wait lists may be associated with the following:

- One or more organization units
- One resource (staff member, resource group, care site, equipment)
- One or more appointment types
- One or more appointment series definitions
- One specialty

Appointment types associated with a wait list must have valid organization appointment types defined at each wait list organization unit specified.

Optionally, a specialty can be associated with a wait list. Because staff members and practice settings can have more than one specialty or sub-specialty, the one pertaining to the particular wait list should be specified. Even if a resource is not associated with the wait list, a specialty can still be identified.

Applications can define a sort order among the following weighting factors to help determine the sequence of entries within a wait list:

- Severity
- Date added to wait list

- Referral date
- Date referral received

4.8.10.2 Creating Wait List Entries

Use the `createWaitListEntry` method to create new wait list entries. For each entry, you can specify:

- One or more patients
- Wait list
- Severity (extensible lookup—seeded values are *urgent*, *soon*, *routine*)
- Referral date, if applicable
- Date referral received, if applicable

Organization units, resource, specialty, appointment types and appointment series are not stored at the wait list entry level, but are associated (when present) through the wait list linked to the entry. You can attach one or more clinical documents to a wait list entry, using HTB Clinical Business Services (Section 4.9), by creating or updating a clinical act to include the wait list entry patient ID.

See Also: Section 4.9, Implementing Clinical Business Services

4.8.10.3 Deleting Wait Lists

Use the `deleteWaitList` method to inactivate wait lists by changing their status to *aborted*. Wait lists that contain active wait list entries cannot be inactivated; the entries must first be transferred or removed.

4.8.10.4 Deleting Wait List Entries

Use the `deleteWaitListEntry` method to inactivate wait list entries by changing their status to *aborted*. You must select an action reason from a list of values when deleting a wait list entry.

4.8.10.5 Finding Wait Lists

Use the `findWaitLists` method to find wait lists that match the search criteria.

4.8.10.6 Finding Wait List Entries

Use the `findWaitListEntries` method to find wait list entries that match the search criteria.

4.8.10.7 Getting Wait Lists

Use the `getWaitList` method to retrieve a single wait list by its wait list identifier.

4.8.10.8 Getting Wait List Entries

Use the `getWaitListEntry` method to retrieve a single wait list entry by its wait list entry identifier.

4.8.10.9 Moving Wait List Entries

Use the `moveWaitListEntries` method to transfer specified entries from their current wait list to the target wait list.

4.8.10.10 Updating Wait Lists

Use the `updateWaitList` method to update an existing wait list. Business rules governing the creation of wait lists also apply to the updating of wait lists. You can edit all wait list attributes, except encounter class, for all wait list statuses except *nullified*. To reactivate a wait list that has been inactivated, you can update the status from *aborted* to *active*. This action only affects the wait list status and does not reestablish any entries that have been transferred or removed.

4.8.10.11 Updating Wait List Entries

Use the `updateWaitListEntry` method to update existing wait list entries. Business rules governing the creation of wait list entries also apply to the updating of wait list entries. All wait list entry attributes can be edited; edits can be made to entries of any status except *aborted* or *nullified*. To suspend, reactivate, complete, or nullify a wait list entry, you can change the action code (which internally updates the status code). You must select an action reason from the list of values each time the action code is modified.

The status code is immediately updated when you run the `updateWaitListEntry` method. The action date reflects the date a status is changed only when the action date is the same as the system date. To accurately calculate wait time using status and action dates, avoid using future action dates. For example, you would suspend a wait list entry on the day the suspension begins.

4.8.11 Implementing Patient Scheduling

Patient Scheduling supports the creation and management of **appointment**s, including the following principal features:

- Appointment Service

- Appointment Type Service
- Appointment Series Definition Service
- Appointment Question Service
- Support for Inbound and Outbound Messaging Services
- Linking **clinical act**s (such as order, referral, procedure, and preliminary diagnosis) to patient appointment by including the patient appointment ID on the clinical act.
- Versioning and version queries: All appointment and dependent tables are versioned. Patient Scheduling Services extends Versioning Services to support the following queries:
  - Point-In-Time
  - Version History
  - Previous Version
  - Next Version
  - Audit (created by, created when)

Reference

Oracle Javadoc for HTB

Table 4–78 lists the principal Patient Scheduling service and methods:

Table 4–78 Service and Methods: Patient Scheduling

Level	Detail
`Package`	`oracle.apps.ctb.admin.scheduling`
`Class`	`AppointmentService`

Table 4–78 (Cont.) Service and Methods: Patient Scheduling

Level	Detail
Methods	■ createAppointment ■ createAppointmentCollection ■ createPatientAppointmentAnswer ■ findAppointmentCollections ■ findAppointments ■ findPatientAppointmentAnswers ■ findPatientAppointments ■ findPatientAppointmentXRefsByInstanceIdentifier ■ getAppointment ■ getAppointmentCollection ■ getPatientAppointmentAnswer ■ getPatientAppointmentByExternalIdentifier ■ performAction ■ performActions ■ updateAppointment ■ updatePatientAppointmentAnswer
Class	AppointmentTypeService
Methods	■ createAppointmentType ■ deleteAppointmentType ■ findAppointmentTypes ■ getAppointmentType ■ updateAppointmentType
Class	AppointmentSeriesDefinitionService
Methods	■ createAppointmentSeriesDefinition ■ deleteAppointmentSeriesDefinition ■ findAppointmentSeriesDefinitions ■ getAppointmentSeriesDefinition ■ updateAppointmentSeriesDefinition
Class	AppointmentQuestionService

Table 4–78 (Cont.) Service and Methods: Patient Scheduling

Level	Detail
Methods	■ createAppointmentQuestion ■ deleteAppointmentQuestion ■ findAppointmentQuestions ■ getAppointmentQuestion ■ updateAppointmentQuestion

Prerequisites

- Implementing Organizations: An organization unit must exist for an appointment or organization appointment type to be created.
- Implementing Resource Management: A resource must exist to be associated with an appointment, organization appointment type or appointment question. Staff members, care sites, equipment and resource groups must be created as resources before they can be associated with an appointment.
- Implementing Patient Registration: A patient with a medical record must exist for appointment of status *active* to be created.
- Implementing Enterprise Terminology Services: ETS must be implemented, to facilitate creation of concept lists.
- Implementing Inbound Messaging Services: Required when using inbound messaging.
- Implementing Outbound Messaging Services: Required when using outbound messaging.
- Implementing Cross-Referencing: Required when using Inbound Messaging Services.
- Creating a Concept List: Values for extensible Patient Scheduling concept lists should be created as required.
- **Defining Concept List Values:**
  - Values for appointment type names must be defined for organization appointment types to be created. Scheduling and Encounter services share the concept list Encounter Type Code, because the same values are used for encounter types and appointment types.

- Values for clinical act definition IDs (such as codes contained in ICD-9, ICD-10, CPT-4, SNOMED, OPCS) and procedure codes must exist to be used as the appointment type code sets that are associated with appointment types. Scheduling populates the procedure code field with the Detail ID of the clinical act, when present. Ideally, you can use the ETS ID for this; it should contain the code value (such as 0012T, 76970,...).
- Values for procedure categories (category code concept list) are used to help organize appointment types. For example, imaging may be the procedure category for appointment types such as echocardiogram and chest x-ray. They must exist to be associated with appointment types.
- Values for Acuity Level Code must exist to be associated with patient appointments. *Note: acuity level is only associated with patient appointments and encounters that are of encounter class Inpatient.*

See Also:

- Section 4.3, Implementing Organizations
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.5.6.1, Creating a Concept List
- Section 4.8.7, Implementing Resource Management
- Section 4.8.9, Implementing Patient Registration
- Section 4.10, Implementing Cross-Referencing
- Section 4.12, Implementing Inbound Messaging Services
- Section 4.13, Implementing Outbound Messaging Services
- Appendix H, Empty Concept Lists
- *HTB Concept Lists Index, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page)*

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal patient scheduling tasks:

- Section 4.8.11.1, Managing Appointment Types
- Section 4.8.11.2, Managing Appointment Series Definitions
- Section 4.8.11.3, Managing Appointment Questions
- Section 4.8.11.4, Managing Appointments
- Section 4.8.11.5, Workflow Setup Recommendations

4.8.11.1 Managing Appointment Types

Use the `createAppointmentType` method to create appointment types. Appointment types provide a template for the duration, description, and resource requirements of the appointment, and the actions to be performed. Examples include allergy shot, follow up consultation, chemotherapy and back physical therapy. The appointment type name is defined using Enterprise Terminology Services. This method (i) associates the name, procedure category (category code), and appointment type code sets (`ActDefIDs` and procedure codes) with the appointment type, and (ii) associates organization-specific attributes with the organization appointment type. Organization appointment types contain attributes unique to each organization.

Resources may be needed for only a portion of an appointment. If the duration a resource is needed is less than the duration defined by the appointment type, include start time of resource and duration for the resource. The application can use this shorter duration when calculating the resource's availability instead of the entire appointment duration.

When the following fields are specified, it causes appointments of this appointment type to be created as non-movable, non-conforming custom appointments. Non-movable appointments are limited to the specific time slot chosen during appointment creation. The application generally does not adjust these types of appointments within a time block to maximize the number of appointments that can occur within that time block. Non-conforming appointments are those that do not fit into a resource's schedule template of available time slots for particular appointment types. Creating several such appointments may limit availability.

- Minutes After Appointment Start Time (start time for a particular resource).

- Resource Duration (must be less than the appointment type Duration less the Minutes After Appointment Start Time prior to the resource's involvement).

Use the following methods to manage appointment types:

- Use the `deleteAppointmentType` method to render an appointment type inactive on the effective end date specified.
- Use the `findAppointmentTypes` method to find appointment types that match the specified search criteria.
- Use the `getAppointmentType` method to retrieve a single appointment type by its appointment type identifier. All attributes are returned.
- Use the `updateAppointmentType` method to update the appointment type with any changed attributes.

4.8.11.2 Managing Appointment Series Definitions

Use the `createAppointmentSeriesDefinition` method to create a new appointment series definition. Appointment series let a user schedule a set of predefined, related appointments, including specified appointment types, with a few steps.

For example, anemic patients may be scheduled for an initial consultation followed by an optional diagnostic blood test on the same day, followed by a blood transfusion within twenty-four hours, followed by a fifteen minute follow-up within seven days.

Prior to the creation of any sequential appointments, the system administrator must configure the appointment series definition along with the appropriate rules that the system must follow in order for the appointment series to be valid.

For example, if there is a minimum or maximum amount of time that can occur between each appointment in the series, these time constraints can be configured (in minutes).

The *autosequence* flag does not constrain the underlying rows. If set to Y, it suggests that appointments can be created in any order, regardless of sequence numbering of the appointment series definition rows.

Each appointment series definition must contain at least two rows.

Use the following methods to manage appointment series definitions:

- Use the `deleteAppointmentSeriesDefinition` method to delete an appointment series definition on the effective end date specified.

- Use the `findAppointmentSeriesDefinitions` method to find appointment series definitions based on specified query criteria.
- Use the `getAppointmentSeriesDefinition` method to retrieve a single appointment series definition by its appointment series definition identifier.
- Use the `updateAppointmentSeriesDefinition` method to update the appointment series definition with any modified information.

4.8.11.3 Managing Appointment Questions

Use the `createAppointmentQuestion` method to create a new appointment question definition and map the question to one or more organization appointment types, organization units or resources. For questions that have answers of type LOV, LOVs can be added at the time of question creation.

Appointment questions are preconfigured questions that are asked of patients during the scheduling process. A question will be associated with any appointments that contain an organization, resource or appointment type to which the question is mapped.

Use the following methods to manage appointment questions:

- Use the `deleteAppointmentQuestion` method to inactivate an appointment question definition on the effective end date.
- Use the `findAppointmentQuestions` method to find appointment question definitions.
- Use the `getAppointmentQuestion` method to retrieve a question by its appointment question definition identifier. All attributes, LOV answers and mappings are returned.
- Use the `updateAppointmentQuestion` method to update an appointment question. If the appointment question is active (the current date is past the start date of the question), the definition of the question, such as name, text, LOVs, cannot be changed. However, question mappings to appointment types, resources and organization units can be added and deleted. A question can only be updated when the question is not yet effective, or the question is effective but only the effective end date is changed (when performing a delete).

  For questions that have answers of type LOV, LOVs are added, removed and updated using this method. The list contained in the `getQuestionLOVs` array becomes the new LOV list after the update method call.

4.8.11.4 Managing Appointments

Use the `createAppointment` method to create a new appointment. Time, duration, encounter class (inpatient, ambulatory, emergency) appointment type, resources and chief complaint are among the attributes that can be specified. If a resource is needed for less than the entire appointment, the resource's start and end time can default from the appointment type or be specified at the time of appointment creation. If no patient is associated with the appointment, the appointment status is set to *reserved*. A successfully created appointment that contains a patient has a status of *active*. Application search availability methods can be used to first determine appointment availability prior to using this method.

An appointment is used to define a single visit. Patient-specific attributes are defined in the patient appointment value object using this method. One or more patients (group appointment) may be associated with an appointment.

The Parent Encounter ID on the patient appointment refers to the inpatient encounter during which the procedure was ordered; it has nothing to do with encounter grouping or linking.

Clinical documents can be attached using Clinical Business Services by creating or updating a clinical act to include the patient appointment ID on the clinical act (Section 4.9).

From the HL7 perspective, an appointment is an encounter in appointment **mood**. For inbound and outbound messaging using HL7 messages, and for applications using HTB, access to appointment data is available only through Patient Scheduling Services.

Use the following methods to manage appointments:

- Use the `createAppointmentCollection` method to create a new appointment collection. Collections can be of type *repeating series* or *appointment series*. Repeating series are not defined in advance. The application can create appointments based on repeating intervals and use this method to group them as a series. Appointment series collections are groupings of appointments that are created using appointment series definitions.
- Use the `createPatientAppointmentAnswer` method to create an answer to an existing question associated with a patient appointment.
- Use the `findAppointmentCollections` method to find appointment collections based on the specified query criteria.
- Use the `findAppointments` method to find appointments based on specified criteria. Search terms `createPatientAppointmentSearchTerm` and

`createAppointmentResourceSearchTerm` can be used to define search criteria using attributes of the `PatientAppointment` and `AppointmentResource` value objects.

- Use the `findPatientAppointmentAnswers` method to find an array of `PatientAppointmentAnswer` value objects based on the specified query criteria.
- Use the `findPatientAppointmentXRefsByInstanceIdentifier` method to get an array of `PatientAppointmentXRef` cross-reference value objects that map an internally generated `PatientAppointment` identifier to an externally assigned instance identifier (root + extension).
- Use the `getAppointment` method to retrieve a single appointment by its appointment identifier. All existing attributes for the appointment are returned.
- Use the `getAppointmentCollection` method to retrieve a single appointment collection by its appointment collection identifier.
- Use the `getPatientAppointmentAnswer` method to retrieve a patient appointment answer by its unique identifier.
- Use the `getPatientAppointmentByExternalIdentifier` method to get a patient appointment identified by the externally assigned instance identifier (root + extension).
- Use the `performAction`(s) methods to define the action(s) that can be performed on an existing appointment. Actions may only modify appointments that have not yet been completed. Use the `performAction` method to perform multiple actions to the same appointment (such as reactivate and reschedule).

  Valid actions include the following:

  – **Nullify:** Invalidates an appointment. Changes the state of all patient appointments in this appointment to *nullified*. Select from the lookup values to provide optional values for action reason.

  – **Cancel:** Cancels an appointment. Changes the state of all patient appointments in this appointment to *aborted*. Select from the lookup values to provide required values for action reason and cancel initiator.

  – **Cancel Appointment Collection:** Cancels an entire collection of appointments. Select from the lookup values to provide required values for action reason and cancel initiator. Individual appointments in the collection can alternatively be cancelled one at a time, using the `performAction: cancel` method.

  - **Complete:** Completes an appointment. Changes the state of all patient appointments in this appointment to *completed*. Use Encounter Service APIs to manage the associated encounters for completed appointments. Appointments with status *reserved* must first have a patient associated before being completed. Select from the lookup values to provide optional values for action reason.
  - **Reactivate:** Reactivates an appointment. Changes the state of all patient appointments in this appointment to *active* or *reserved* (if a patient is not associated). Select from the lookup values to provide required values for action reason.
  - **Reschedule:** Reschedules an appointment. Changes the date, time, or resources of an appointment. Select from the lookup values to provide required values for action reason.

- Use the `updateAppointment` method to add, remove or modify patients, and to update appointment attributes including:
  - Chief complaint
  - Referrer name
  - Referrer phone number
  - Procedure codes
  - Special requests
  - Notes
- Use the `updatePatientAppointmentAnswer` method to update the answer to a question associated with a patient appointment.

4.8.11.5 Workflow Setup Recommendations

You can configure Oracle Workflow to use the `performAction: cancel` method to automatically cancel an appointment and notify any associated resources, if no patients are scheduled within a configurable amount of time from the appointment start time. The workflow administrator can configure the timeout process by indicating the number of minutes (positive if before, negative if after) relative to the appointment start time. You can also define the frequency and start time for the work flow.

4.8.12 Implementing Encounter Management

Encounter Management provides a mechanism for creating and managing an **encounter** between a practitioner and a patient, at a specific location for a specified time period. Encounter records contain administrative, financial, and clinical data related to one or more encounters for each patient, including the following encounter types:

- **Inpatient:** Encounters requiring an admission to an acute care facility for at least 24 hours.
- **Outpatient:** Encounters of less than 24 hours not requiring a bed.
- **Emergency:** Emergent healthcare services provided by specific facilities and practitioners.

Encounters can be organized by **encounter group**, which can contain individual encounters or other encounter groups. Examples of encounter groups include:

- Clinical studies
- Episodes of care
- Group therapy
- Recurring treatment

Two encounters, for the same or different patients, can be linked when they have a specific relationship (clinical, administrative, financial...). Examples of encounter links include:

- Donor to recipient
- Mother to newborn
- Preadmit testing in advance of a planned procedure
- Emergency visit to subsequent inpatient admission

Reference

Oracle Javadoc for HTB

Table 4–79 lists the principal Encounter Management service and methods:

Table 4–79 Service and Methods: Encounter Management

Level	Detail
Package	`oracle.apps.ctb.admin.encounter`

Table 4–79 (Cont.) Service and Methods: Encounter Management

Level	Detail
Class	`EncounterService`
`Methods`	▪ `createEncounter` ▪ `createEncounterGroup` ▪ `findEncounterGroups` ▪ `findEncounters` ▪ `findEncounterXRefsByInstanceIdentifier` ▪ `getEncounter` ▪ `getEncounterByExternalIdentifier` ▪ `getEncounterGroup` ▪ `performAction` ▪ `performActions` ▪ `updateEncounter` ▪ `updateEncounterGroup`

Prerequisites

- **Implementing Patient Registration:** A patient with a medical record must exist.
- **Implementing Organizations:** A practice setting and facility must exist.

See Also:

- Section 4.3, Implementing Organizations
- Section 4.8.9, Implementing Patient Registration

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.10.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal tasks related to encounter management:

- Section 4.8.12.1, Creating Encounters
- Section 4.8.12.2, Finding Encounters
- Section 4.8.12.3, Getting Encounters
- Section 4.8.12.4, Performing Actions
- Section 4.8.12.5, Updating Encounters
- Section 4.8.12.6, Creating Encounter Groups
- Section 4.8.12.7, Finding Encounter Groups
- Section 4.8.12.8, Updating Encounter Groups
- Section 4.8.12.9, Linking Encounters
- Section 4.8.12.10, Cancelling Encounters
- Section 4.8.12.11, Encounter Bed Management

4.8.12.1 Creating Encounters

Use the `createEncounter` method to create new encounters. When creating an encounter, an *initial event* must also be created; this determines the initial state of the encounter. Initial events include the following:

- **Preadmit event:** An encounter without a confirmed appointment date.
- **Pending-admit event:** An encounter with a confirmed appointment date.
- **Admit event:** An encounter that commences immediately.

Encounter events are thus validated against the encounter status. For example, you cannot create an *admit* encounter event for a terminated encounter.

An encounter is classified as one of the following:

- Inpatient Stay
- Emergency Visit
- Outpatient Visit

Encounter attributes are defined by the `Encounter` value object.

4.8.12.2 Finding Encounters

User the `findEncounters` method to find encounters of a single class or all classes. For single class searches there are different search criteria for inpatient stays, outpatient visits, and emergency visits. For searches across encounter classes there are summary encounter search criteria.

Given an encounter identifier or other search criteria, you can use the `findEncounters` method to find accompanying persons, or you can use the `findEncounterXRefsByInstanceIdentifier` method to find a CTB encounter identifier based on an external identifier.

4.8.12.3 Getting Encounters

Use the `getEncounterByExternalIdentifier` method to get an externally identified encounter.

4.8.12.4 Performing Actions

Use the `performAction` method to perform a specific action on an encounter. Use the `performActions` method to perform more than one action on an encounter. For either method you can use either the encounter or encounter ID as a parameter. You can perform the following actions:

- Assign location (`AssignLocationAction`)
- Cancel (`CancelAction`)
- Entered in error (`EnteredInErrorAction`)
- The following actions can be performed for inpatient stays:
  - Admit (`InpatientStayAdmitAction`)
  - Decease Patient (`InpatientStayDeceasePatientAction`)
  - Discharge (`InpatientStayDischargeAction`)
  - Leave (`InpatientStayLeaveAction`)
  - Pending Admit (`InpatientStayPendingAdmitAction`)
  - Return From Leave (`InpatientStayReturnFromLeaveAction`)
  - Swap Care Site (`InpatientStaySwapCareSiteAction`)
  - Transfer Attending Physician (`InpatientStayTransferAttendingPhysicianAction`)
  - Transfer Location (`InpatientStayTransferLocationAction`)

  - Transfer Medical Service (`InpatientStayTransferMedicalServiceAction`)
  - Transfer Type (`InpatientStayTransferTypeAction`)
- The following actions can be performed for outpatient and emergency visit actions:
  - Checkin (`VisitCheckinAction`)
  - Checkout (`VisitCheckoutAction`)

    See Also: Interface `Action` (`oracle.apps.ctb.admin.encounter`) for more information about actions available to the `performAction` and `performActions` methods.

4.8.12.5 Updating Encounters

Use the `updateEncounter` method to update existing encounters. Business rules governing the creation of encounters also apply to the updating of encounters.

4.8.12.6 Creating Encounter Groups

Use the `createEncounterGroup` method to create new encounters. Encounter groups contain many of the same attributes that can be applied at the individual encounter level. Encounters or other encounter groups can participate as members of an encounter group, thereby supporting a hierarchy of encounters. A single encounter group's members can include a combination of individual encounters and encounter groups.

Encounter group attributes are defined by the `EncounterGroup` value object.

4.8.12.7 Finding Encounter Groups

Use the `findEncounterGroups` method to find encounter groups, using search attributes defined by the `EncounterGroupQueryCriteria` interface. Use the `findEncounters` method to search for group members or confidentiality codes associated with a group.

4.8.12.8 Updating Encounter Groups

Use the `updateEncounterGroup` method to update an encounter group, including its members.

4.8.12.9 Linking Encounters

Encounter links are used to create a relationship between two encounters, such as donor/recipient or mother/child. Use the `setEncounterLinks` method in the `Encounter` interface to create encounter links. Link attributes are defined by the `EncounterLink` value object.

4.8.12.10 Cancelling Encounters

Use the `cancelEncounter` method to cancel an existing encounter, by changing its status to *cancelled* and rendering it inactive. Cancelling an encounter event is subject to the status transition rules of encounter events. The last encounter event within an encounter must be cancelled before the encounter itself can be cancelled.

4.8.12.11 Encounter Bed Management

Encounter Management and Care Site Management jointly provide a bed management mechanism for a healthcare organization. The `CareSiteService` interface provides some of the methods.

However, the daily operations of bed management are distinguished from the administrative creation and maintenance of care sites. Patient movements in and out of care sites are consequences of encounter actions, and subsequent activities such as housekeeping and maintenance are thus handled as part of bed management. Such activities include the following:

- Reserving Care Sites
- Cancelling Care Site Reservations
- Requesting Care Site Housekeeping or Maintenance Activity
- Updating Care Site Housekeeping or Maintenance Activity

Reserving Care Sites

Use the `reserveCareSite` method to reserve a care site from the current date forward. The care site can be reserved in anticipation of a pending encounter, or for a specific encounter. The care site status changes to `STATUS_RESERVED`.

Cancelling Care Site Reservations

Use the `cancelCareSiteReservation` method to cancel an existing care site reservation. This clears out the reserved reason and reserved from date. The care site status changes to `STATUS_EMPTY`.

Requesting Care Site Housekeeping or Maintenance Activity

Use the `requestCareSiteActivity` method to request housekeeping or maintenance for a care site (for example, after a patient has checked out). Specify whether the activity type is Housekeeping or Maintenance and a comment. The activity request date defaults to the current date and time, and the Housekeeping/Maintenance flag is set to Yes.

Updating Care Site Housekeeping or Maintenance Activity

Use the `careSiteActivityComplete` method to indicate that a requested care site activity is complete. Specify the care site identifier and whether the activity type is Housekeeping or Maintenance. The activity request date and comment are reset to *null*.

4.8.13 Implementing Patient List Management

Patient List Management maintains the foundational data set necessary to support creation, maintenance and retrieval of patient lists within a healthcare organization. Patient lists are routinely viewed by organization staff members on a daily basis, with access based on list ownership (resource groups or individual users). Resource groups are administrative groupings of staff and resources that can include staff, equipment and care sites.

Patient lists are encounter-based or appointment-based. Encounter-based lists have a list type of *Staff*, *Location Medical Service*, or *Custom*; appointment-based lists have a list type of *Appointment*.

All patient lists are patient-centric. They can have multiple filters and have at least one attribute. List attributes are configurable by the list owner, who can specify a default list using profile options.

Patient List methods (Table 4–80) let you perform the following functions:

- Create, update and retrieve patients lists.
- Find patient lists.
- Add and remove columns from patient lists.

Reference

Oracle Javadoc for HTB

Table 4–80 lists the principal Patient List service and methods:

Table 4–80 Service and Methods: Patient List

Level	Detail
Package	`oracle.apps.ctb.admin.patient.list`
Class	`PatientListService`
Methods	▪ `createPatientLists` ▪ `findPatientLists` ▪ `getListValues` ▪ `getPatientList` ▪ `updatePatientLists`

Prerequisites

Implementing all prior Administrative Business Services (generates records upon which patient lists are based).

See Also:

- Section 4.8.1, Implementing Payers and Plans
- Section 4.8.2, Implementing Employers
- Section 4.8.3, Implementing Address Management
- Section 4.8.4, Implementing Staff Management
- Section 4.8.5, Implementing Care Sites
- Section 4.8.6, Implementing Equipment Management
- Section 4.8.7, Implementing Resource Management
- Section 4.8.8, Implementing Workgroups
- Section 4.8.9, Implementing Patient Registration
- Section 4.8.10, Implementing Wait List Management
- Section 4.8.11, Implementing Patient Scheduling
- Section 4.8.12, Implementing Encounter Management
- Section 4.8.13, Implementing Patient List Management

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

The following sections describe the principal patient list tasks:

- Section 4.8.13.1, **Creating Lists**
- Section 4.8.13.2, Finding and Retrieving Lists
- Section 4.8.13.3, **Updating Lists**

4.8.13.1 Creating Lists

Use the `createPatienttLists` method to create a new patient list. The list can be of type Location, Appointment, Medical Service, or Custom. The main patient list interface consists of common attributes, such as columns, name of list, patients on the list, start and end dates. The list types extend the main patient list interface and provide additional filters on which to base the patient list (Table 4–81):

Table 4–81 Patient List: Filters by List Type

List Type	Filter
Appointment	▪ Appointment State ▪ Appointment Time ▪ Facility ▪ Practice Setting ▪ Resource ▪ Resource Group ▪ Resource Type
Custom	▪ Date ▪ Encounter State ▪ Patient

Table 4–81 (Cont.) Patient List: Filters by List Type

List Type	Filter
Location	■ Encounter Class ■ Encounter State ■ Facility ■ Practice Setting
Medical Service	■ Encounter Class ■ Encounter State ■ Facility ■ Medical Service ■ Practice Setting
Staff	■ Encounter Class ■ Encounter Type ■ Facility ■ Practice Setting ■ Resource ■ Resource Group ■ Resource Type ■ Role Code

Specifying the workgroup identifier lets staff members of the workgroup view the data on the list.

See Also: Section 4.8.8, Implementing Workgroups

Appointment

Use the *Appointment* type to retrieve patients who have a scheduled appointment (**encounter** appointment) in a specified **practice setting** on a specific date—optionally qualified by who the patient is scheduled to see (an individual or a resource group).

Appointments include the following:

- Patients with scheduled appointments in a specified practice setting.
- Patients with appointments in a specified practice setting on a specific date.
- Patient appointments for a specified **health care provider** or group.

Location

Use the *Location* type to retrieve all patients associated with a specified facility or practice setting.

Location includes the following:

- Inpatients in a specified facility *or* practice setting with encounters in a specified state.
- Outpatients in a specified facility *or* practice setting with an active encounter.
- Inpatients or outpatients in a specified facility *and* practice setting.

Medical Service

Use the *Medical Service* type to retrieve all patients in the enterprise associated with a specific medical service. The medical service list can include patients who have **inpatient** or **outpatient** encounters. The state of the encounter can be *active* or *inactive*.

Staff

Use the *Staff* type to retrieve all patients associated with a specific staff member. The staff list can include patients that have inpatient or outpatient encounters, and the state of the encounters can be active or inactive. It can also include patients at a specific location or those requiring a specific resource.

Custom

Use the *Custom* type to retrieve all patients in a specified custom list. The **clinician** identifies specific individual patients and adds them to this type of list. For example, a clinician might set up such a list to follow up on patients that are problematic, to organize patients by recency of encounters or visits, or to identify patients with a common disease process requiring continuing preventive care.

4.8.13.2 Finding and Retrieving Lists

- Use the `findPatientLists` method to find patient lists that match search criteria; returns an array of patient list objects.
- Use the `getPatientList` method to retrieve a patient list by its unique identifier.
- Use the `getListValues` method to retrieve an array of patient list row value objects; returns a single patient list row for each patient that matches the list filter.

4.8.13.3 Updating Lists

Use the `updatePatientLists` method to update an existing patient list. Business rules governing the creation of patient lists also apply to the updating of patient lists.

4.9 Implementing Clinical Business Services

Clinical Business Services provides structured access to the HTB data repository for healthcare clinical management activities. These services provide a foundation for the development of healthcare applications and support the recording and retrieval of clinical data, including the following activities:

- Recording or retrieving a diagnosis for a patient by a clinician.
- Recording or retrieving orders for tests and exams, or recording and reviewing results of such tests
- Recording or retrieving a dietary order, or confirmation that a diet order has been carried out.
- Recording or retrieving a procedure order, or confirmation that a procedure has been performed.
- Recording or retrieving medication orders (substance administration), or confirmation that such orders have been carried out.
- Recording or retrieving a supply order, or confirmation that a medical supply has been dispensed.

> **See Also:** Table K–1 (Appendix K), for a description of inbound messages that contain clinical data.

Reference

Clinical Business Services supports the persistence, retrieval, and management of clinical data, including methods that facilitate retrieval and management of individual clinical data instances. These services are grouped into three packages:

- Master Catalog
- User Catalog
- Clinical Acts

Table 4–82 lists the principal packages and service interfaces:

Table 4–82 Clinical Packages and Service Interfaces

Package	Service Interface
`oracle.apps.ctb.clinical.act`	▪ `ClinicalActService`
`oracle.apps.ctb.clinical.mastercatalog`	▪ `ActDefinitionService` ▪ `ActStateTransitionService`

Table 4–82 (Cont.) Clinical Packages and Service Interfaces

Package	Service Interface
oracle.apps.ctb.clinical.usercatalog	▪ UserCatalogService

The Clinical Act Service supports the creation, query, and update of clinical act instances. The Master Catalog is a collection of Act Definitions that facilitate querying and template generation at a healthcare organization. The User Catalog represents a customer-defined subset of the Master Catalog that can be used to manage templates.

There are two ways that clinical data is received by HTB: (i) Inbound messages from external systems, and (ii) applications that act as HTB clients. Clinical Business Services is used for persisting and managing clinical data, irrespective of the source.

Prerequisites

- Implementing Security Services
- Implementing Enterprise Terminology Services
- Implementing Audit Services
- Implementing Administrative Business Services

See Also:

- Section 4.1, Implementing Security Services
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.6, Implementing Audit Services
- Section 4.8, Implementing Administrative Business Services

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Code Examples

Table 4–83 summarizes code samples used in the Clinical Business Services implementation process:

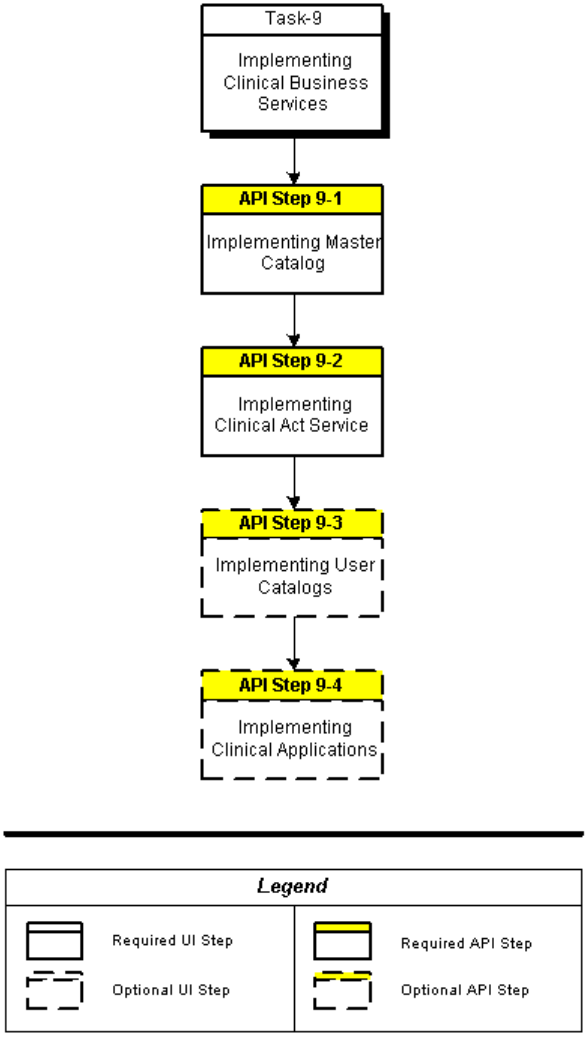
Table 4–83 Clinical Business Services: Code Examples

Example Number	Description
4–8	Code Sample: Create an Act Definition
4–9	Code Sample: Finding an Act Definition
4–10	Code Sample: Creating a new Medication Order using DBEV
4–11	Code Sample: Update Clinical Acts with DBEV
4–12	Code Sample: Creating a User Catalog
4–13	Code Sample: Creating a User Catalog Folder
4–14	Code Sample: Creating a Medication Template
4–15	Code Sample: Creating a Clinical Act Template Link
4–16	Code Sample: Retrieving a Medication Template in a User Catalog
4–17	Code Sample: Demographic Display
4–18	Code Sample: Encounter Display
4–19	Code Sample: List User Catalogs where the logged-in user is a member of an associated Workgroup
4–20	Code Sample: Using ClinicalActService
4–21	Code Sample: Finding all Active Substance Administration Orders for a Specified Encounter
4–22	Code Sample: Retrieving Knowledge Document information

Procedures

Figure 4–20 provides an overview of the implementation process for Clinical Business Services:

Figure 4–20 Implementation Process: Clinical Business Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–10, HTB Implementation Procedures: Clinical Business Services

The following sections describe the implementation procedures for Clinical Business Services (referenced by Figure 4–20), including two illustrative applications:

- Section 4.9.1, Implementing Master Catalogs
- Section 4.9.2, Implementing Clinical Act Service
- Section 4.9.3, Implementing User Catalogs
- Section 4.9.4, Implementing Clinical Applications
- Section 4.9.5, Implementing a Medication Order Entry System
- Section 4.9.6, Implementing a Results Reviewing System

4.9.1 Implementing Master Catalogs

The Master Catalog is a required HTB component that must be implemented before using HTB. It facilitates creation, authorization, and retrieval of clinical data, and provides services for the creation and management of clinical **Act Definition**s—high-level categories of individual **clinical act**s that an organization elects to record. Organizations implementing the Master Catalog create act definitions to categorize individual clinical acts that they want to record.

Every instance of a clinical act must be associated with a clinical act definition. Act definitions group Enterprise Terminology Service (ETS) concepts into meaningful categories such as Substance Administration Orders and Observation Orders.

See Also: Section 4.5 for more information about ETS.

Clinical Act Definitions

Each of the following Act Definition attributes must be specified:

- Class

- Mood
- Category Code
- Detail Type Code
- Organization Identifier
- Effective Start Time

Table 4–84 lists valid combinations of class and mood (Act Types) that are preseeded in HTB:

Table 4–84 Predefined Act Types

Class	Mood
Observation (OBS)	Request for Order (RQO)
Observation (OBS)	Event (EVN)
Observation (OBS)	Event Criterion (EVN.CRT)
Supply (SPLY)	Request for Order (RQO)
Supply (SPLY)	Event (EVN)
Supply (SPLY)	Event Criterion (EVN.CRT)
Diet (DIET)	Request for Order (RQO)
Diet (DIET)	Event (EVN)
Diet (DIET)	Event Criterion (EVN.CRT)
Procedure (PROC)	Request for Order (RQO)
Procedure (PROC)	Event (EVN)
Procedure (PROC)	Event Criterion (EVN.CRT)
Substance Administration (SBADM)	Request for Order (RQO)
Substance Administration (SBADM)	Event (EVN)
Substance Administration (SBADM)	Event Criterion (EVN.CRT)
Patient Care Provision Referral (PCPR)	Request for Order (RQO)
Patient Care Provision Referral (PCPR)	Event (EVN)
Patient Care Provision Referral (PCPR)	Event Criterion (EVN.CRT)
Condition (COND)	Event (EVN)
Condition (COND)	Event Criterion (EVN.CRT)
Public Health Case (CASE)	Event (EVN)
Public Health Case (CASE)	Event Criterion (EVN.CRT)

There is a validation mechanism for Act Definitions; the following validations must be true:

- If the detail type code is set to *null*; the combination of class code, mood code, organization ID, and category code must be unique.
- If the detail type code is set to *Any*; only one act definition with the same class code, mood code, organization ID and category code with a null detail ID is allowed.
- If the detail type code is set to ID; the combination of Class Code, Mood Code, Organization Id, Category Code and Detail Id must be unique. There must be no preexisting Act Definitions with the same Class Code, Mood Code, Organization Id, Category Code, and a Detail Type Code of *null*.
- An act definition is valid for all dates between the effective start date and the effective end date. The effective end date can be *null*, in which case the act definition is valid for all dates after the effective start date.

Table 4–85 provides Act Definition examples:

Table 4–85 Act Definition Examples

Class[1]	Mood[1]	Category Code[1]	Detail Type Code[1]	Detail ID[2]	Organization ID[1]	Confidentiality Code[2]
Observation (OBS)	Request for Order (RQO)	Clinical Lab (LAB)	ID	Serum Glucose ETS ID 1234	Enterprise ID	...
Observation (OBS)	Request for Order (RQO)	Clinical Lab (LAB)	ID	HIV ETS ID 678	Enterprise ID	HIV
Observation (OBS)	Event (EVN)	Clinical Lab (LAB)	ANY	...	Enterprise ID	...
Observation (OBS)	Event (EVN)	Diagnostic Image (DXIMG)	ANY	...	Enterprise ID	...
Observation (OBS)	Event (EVN)	Radiology Report (RAD)	NULL	...	Enterprise ID	...
Observation (OBS)	Event Criterion (EVN.CRT)	Precondition (PRECONDITION)	NULL	...	Enterprise ID	...
Supply (SPLY)	RQO	Durable Medical Equipment (DME)	ID	Crutches ETS ID 509	Enterprise ID	...

Table 4–85 (Cont.) Act Definition Examples

Class[1]	Mood[1]	Category Code[1]	Detail Type Code[1]	Detail ID[2]	Organization ID[1]	Confidentiality Code[2]
Supply (SPLY)	EVN	Clinical Drug (CLIN_DRUG)	ANY	...	Enterprise ID	...
Diet (DIET)	EVN	Snack (SNACK)	ANY	...	Enterprise ID	...
Procedure (PROC)	RQO	Surgical Procedure (SURGPROC)	ID	Triple Coronary Artery Bypass Graft ETS ID 187	Enterprise ID	...
Procedure (PROC)	EVN	Surgical Procedure (SURGPROC)	ANY	...	Enterprise ID	...
Procedure (PROC)	EVN	Surgical Procedure (SURGPROC)	ANY	...	Enterprise ID	...
Public Health Case (CASE)	EVN	Notifiably Condition (NOTIFY COND)	NULL	...	Enterprise ID	...
Substance Administration (SBADM)	RQO	Clinical Drug (CLIN_DRUG)	ID	Digoxin ETS ID 0934	Enterprise ID	...
Substance Administration (SBADM)	EVN	Clinical Drug (CLIN DRUG)	ANY	...	Enterprise ID	...
Substance Administration (SBADM)	EVN	Base Drug (BASE CLIN DRUG)	ANY	...	Enterprise ID	...
Substance Administration (SBADM)	EVN	Additive Clinical Drug (ADD CLIN DRUG)	ANY	...	Enterprise ID	...
Substance Administration (SBADM)	EVN	Medication History (MED HX)	ANY	...	Enterprise ID	...
Patient Care Provision Referral (PCPR)	ROQ	Referral (REFER)	...	...	Enterprise ID	...
Condition (COND)	EVN	Problem (PROBLEM)	NULL	...	Enterprise ID	...

[1] Required.

[2] Optional.

Reference

Oracle Javadoc for HTB

Table 4–86 lists the principal methods used to implement the Master Catalog:

Table 4–86 Service and Methods: Implementing the Master Catalog

Level	Detail
Package	`oracle.apps.ctb.clinical.mastercatalog`
Class	`ActDefinitionService`
Methods	▪ `createActDefinition` ▪ `findActDefinitions` ▪ `findActTypes` ▪ `updateActDefinition`

Prerequisites

- Implementing Organizations
- Implementing Enterprise Terminology Services
- Implementing Concept Lists: Necessary Concept Lists must be created

See Also:

- Section 4.3, Implementing Organizations
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.5.6, Implementing Concept Lists
- Appendix B, ETS Supported Terminologies and Cross Maps

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use an appropriate responsibility defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

Perform the following steps to implement and update the Master Catalog:

1. **Create a comprehensive list of clinical acts required by the enterprise:** Analyze this list to identify which terminology best meets enterprise requirements. In some cases, a single standard terminology may be sufficient for a logical group. For example, the LOINC terminology can be used for all laboratory results.

   In other cases, you may have to extend a standard terminology. For example, LOINC+ a local vocabulary could be used for laboratory orders.

   Another case that could occur is when an unmapped local vocabulary scheme is created. This may be required for unique needs such as patient education.

   > **See Also:** Section 4.5 for more information about concept mapping.

2. **Organize (categorize) the list of clinical acts into Act Definitions and define the Act Definition attributes:** For each clinical act identified in step 1, document the act definitions that are required to support the clinical acts in that domain. Act Definitions can only be created with valid combinations of class and mood as defined by preseeded act types. Each of the following Act Definition attributes must be specified:

   - Class
   - Mood
   - Clinical Category Code
   - Detail Type Code
   - Detail Identifier (If Applicable)
   - Organization Code
   - Effective Time

   > **See Also:** Oracle Javadoc for HTB [`oracle.apps.ctb.masterCatalog`] for additional requirements.

- **Confidentiality code:** A key attribute that can be used to create data-level security for a specific clinical Act Definition. For example, if an Act Definition was created for the laboratory test HIV, you may want to create a security rule that prevents some users from viewing any clinical acts that have HIV test details.

  See Also: Table 4–85 for an example of this type of Act Definition.

3. **Load terminologies into ETS:** Load the core, standard, and local terminologies into ETS that are required for creating the enterprise Master Catalog. Also load the relevant cross maps and equivalence maps. This includes mapping the individual clinical acts to standard concepts as appropriate. A type of mapping that could be performed is *equivalent mapping*—in which one concept is defined as the logical equivalent of another.

   See Also: For more information about concept mapping:

   - Section 4.5, Implementing Enterprise Terminology Services
   - Section 4.5.7, Implementing Interterminology and Intraterminology Equivalence

4. **Search ETS for concepts to be used in act definitions:** Use the ETS browser to search for clinical concepts that used in act definitions. For example, if you plan to create an act definition for a lab order for serum glucose, you must search for and note the concept code, coding scheme, and version for use in the act definition.

5. **Build and manage Act Definitions:** Use the Act Definition Service interface to create, maintain, and update act definitions. This is used with clinical and ETS APIs to build master catalog content. It may be helpful to build a spreadsheet with the column headers provided in Table 4–87 to facilitate loading Act Definitions:

Table 4–87 Column Headers

Column Header (Attribute Name)	First Row Values
Description	Ampicillin Sodium
ClassCode	SBADM
MoodCode	RQO
CategoryCode	CLIN_DRUG

Table 4–87 (Cont.) Column Headers

Column Header (Attribute Name)	First Row Values
DetailID	ETS ID (derived from concept code, coding scheme, and version)
Concept Code	00001433-MEDNM
Coding Scheme	FDB
Version	Default
DetailTypeCode	DETAIL_TYPE_ID
EffectiveStartDate	20030901
EffectiveEndDate	...
OrganizationID	ORG12
ConfidentialityCode	...

Example 4–8 Code Sample: Create an Act Definition

```
MastercatalogHelper helper = new MastercatalogHelper();

// create an ActDefinition with a DetailTypeCode of ID
ActDefinition actDefinition = helper.newActDefinition();
actDefinition.setClassCode("SBADM");
actDefinition.setMoodCode("RQO");
actDefinition.setCategoryCode("CLIN_DRUG");
actDefinition.setDetailId("CON-12345");
actDefinition.setOrganizationId("Org123");
actDefinition.setDetailTypeCode(ActDefinition.DETAIL_TYPE_ID);
actDefinition.setEffectiveStartDate(new Date());
ActDefinition createdActDefinition =
actDefinitionService.createActDefinition(actDefinition);

// create an ActDefinition with a DetailTypeCode of ANY
actDefinition = helper.newActDefinition();
actDefinition.setClassCode("SBADM");
actDefinition.setMoodCode("RQO");
actDefinition.setCategoryCode("CLIN_DRUG");
actDefinition.setOrganizationId("Org234");
actDefinition.setDetailTypeCode(ActDefinition.DETAIL_TYPE_ANY);
actDefinition.setEffectiveStartDate(new Date());
createdActDefinition = actDefinitionService.createActDefinition(actDefinition);

// create an ActDefinition with a DetailTypeCode of NULL
actDefinition = helper.newActDefinition();
```

```
actDefinition.setClassCode("COND");
actDefinition.setMoodCode("EVN");
actDefinition.setCategoryCode("PROBLEM");
actDefinition.setOrganizationId("Org345");
actDefinition.setDetailTypeCode(ActDefinition.DETAIL_TYPE_NULL);
actDefinition.setEffectiveStartDate(new Date());
createdActDefinition = actDefinitionService.createActDefinition(actDefinition);
```

6. **Search for an act definition using the `ActDefinitionQueryCriteria` method:** Table 4–88 specifies parameters required to search for a substance administration order act definition.

Table 4–88 Act Definition Query Parameters & Examples

Query Parameters	Example Value
Class	SBADM (Substance Administration)
Mood	RQO (Request for Order)
Category Code	CLIN_DRUG (Clinical Drug)
Detail Id	4568 (description: Furosemide)
Detail Type Code	DETAIL_TYPE_ID
Effective Start Date	20031010 (Oct 10, 2003)
Organization	123 (Id for Memorial Hospital)

The code sample in Example 4–9 illustrates how to search for act definitions. The query is searching for all act definitions in the master catalog that match the specified class, mood, category code and organization values.

Example 4–9 Code Sample: Finding an Act Definition

```
MastercatalogHelper helper = new MastercatalogHelper();

ActDefinitionQueryCriteria qc = helper.newActDefinitionQueryCriteria();
SearchTerm stClass = qc.createSearchTerm(ActDefinitionQueryCriteria.CLASS_CODE,
"SBADM");
SearchTerm stMood = qc.createSearchTerm(ActDefinitionQueryCriteria.MOOD_CODE,
"RQO");
SearchTerm stCategory = qc.createSearchTerm(ActDefinitionQueryCriteria.CATEGORY_
CODE, "CLIN_DRUG");
SearchTerm stOrg = qc.createSearchTerm(ActDefinitionQueryCriteria.ORGANIZATION_
```

```
IDENTIFIER, "123");

// need to combine the Search Terms and set the root Search Term of the Query
SearchTerm stClassMood = qc.and(stClass, stMood);
SearchTerm stCategoryOrg = qc.and(stCategory, stOrg);

qc.setRootSearchTerm( qc.and(stClassMood, stCategoryOrg) );
ActDefinition[] foundDefinitions = actDefinitionService.findActDefinitions(qc);
```

Clinical Business Services include APIs for retrieving and updating existing act definitions.

See Also:

- Interface `ActDefinition` and `ActDefinitionService`, *Oracle Javadoc for HTB, and* Table 4–86
- Appendix K, Act Definition Messaging Considerations
- Appendix L, Clinical Business Services and ETS Concept Lists

4.9.2 Implementing Clinical Act Service

Clinical Acts are the building blocks for recording clinical activities at a healthcare organization. The Clinical Act Service provides methods for creating, updating, and finding Clinical Act instances. It also supports linking of clinical acts to other HTB services, such as persons and materials.

Clinical Act Service methods also provide support for a special type of act called a Clinical Act Template. The creation of Clinical Act Template is discussed in Section 4.9.3.

This section describes how to create a clinical act.

Reference

Oracle Javadoc for HTB

Table 4–89 lists the principal methods used to create, find, and update clinical act instances:

Table 4–89 Service and Methods: Implementing Clinical Acts

Package	Detail
Package	`oracle.apps.ctb.clinical.act`
Class	`ClinicalActService`

Table 4–89 (Cont.) Service and Methods: Implementing Clinical Acts

Package	Detail
Methods	■ createActInstance ■ createTemplates ■ findAlerts ■ findClinicalActs ■ findClinicalActXRefByII ■ findMaterialXRefByII ■ findTemplates ■ getActInstanceDetail ■ getActInstanceHistory ■ getClinicalActByExternalId ■ getKnowledgeDocumentWithDBEV ■ getMaterialByExternalId ■ getMessageContext ■ updateActInstance ■ updateClinicalActDocuments ■ updateTemplates

Prerequisites

- Implementing Organizations
- Implementing Enterprise Terminology Services
- Implementing Concept Lists
- Implementing Master Catalogs

See Also:

- Section 4.3, Implementing Organizations
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.5.6, Implementing Concept Lists
- Appendix B, ETS Supported Terminologies and Cross Maps
- Section 4.9.1, Implementing Master Catalogs
- Appendix L, Clinical Business Services and ETS Concept Lists

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

1. To create a clinical act instance, determine the following:
   - The clinical activity to be represented and supported (such as a medication order). This includes identifying the act definition and specifying the values for any attributes associated with the clinical act. Use the `findActDefinition` method to determine the act definition by searching the master catalog for the act definition associated with the prospective clinical act—*required*.

     See Also: Example 4–9 for a code sample.

   - The participants in the clinical activity (such as a patient [*required*] and a provider [*optional*]).
   - The time when the activity was performed or will be performed—*optional*.
   - The reason why the activity is taking place (the associated clinical acts that necessitated the clinical act being created)—*optional*.
2. Use the accessor methods of `ClinicalAct` to specify values for the substance administration act attributes (step 1).
3. Use the `createActInstance` method to create a clinical act instance, such as a substance administration order.

Example 4–10 Code Sample: Creating a new Medication Order using DBEV

```
String patientId = "12345";
ActHelper helper = new ActHelper();
SubstanceAdministrationAct medicationAct =
```

```
helper.newSubstanceAdministrationAct();
medicationAct.setActDefinitionId("1");
medicationAct.setPatientId(patientId);
medicationAct.setMedicalRecordNumber("1");
medicationAct.setMrnOrgPartyId("123");
medicationAct.setStatusCode("active");
medicationAct.setOperationAsInsert();

SubstanceAdministrationAct partOfMedicationAct =
helper.newSubstanceAdministrationAct();
partOfMedicationAct.setActDefinitionId("1");
partOfMedicationAct.setPatientId(patientId);
partOfMedicationAct.setMedicalRecordNumber("1");
partOfMedicationAct.setMrnOrgPartyId("123");
partOfMedicationAct.setStatusCode("active");
partOfMedicationAct.setOperationAsInsert();

The act relationship code snippet is an optional task depicting the ability to
relate clinical acts to one another:
ActRelationship actRelationship = helper.newActRelationship();
actRelationship.setTypeCode("COMP");  // component relationship
actRelationship.setTargetAct(partOfMedicationAct);
medicationAct.setActRelationships(new ActRelationship[]{actRelationship});

DBEVDocument dbevDocument = new DBEVHelper().newDBEVDocument();

// Set mandatory constraints
dbevDocument.setTargetId(patientId);
dbevDocument.setTargetTypeCode("Patient");
dbevDocument.setSendingSystemCode("DBEV");
dbevDocument.setOwnerId(patientId);
dbevDocument.setContents(
new VAOInterfaceBase[]{medicationAct, partOfMedicationAct});
dbevDocument = mDBEVService.createDBEVs(
new DBEVDocument[]{dbevDocument})[0];

// Submit for approval
mDBEVService.submitDBEVs(new String[]{dbevDocument.getId()});
// Approve ( commit );
mDBEVService.approveDBEVs(new String[]{dbevDocument.getId()});
```

Updating a Clinical Act

To find and update a clinical act instance:

- Use the `findClinicalActs` method to find a clinical act.

  See Also: The following code samples:

  - Example 4–20, Code Sample: Using ClinicalActService
  - Example 4–21, Code Sample: Finding all Active Substance Administration Orders for a Specified Encounter

- Use the `updateActInstance` to update the clinical act

Example 4–11 assumes that the clinical act to be updated is available:

Example 4–11 Code Sample: Update Clinical Acts with DBEV

```
String patientId = "12345";

// we want to set the status of this to completed
existingAct.setStatusCode("completed");
// and also set some other attributes of the Act
existingAct.setEffectiveEndTime(new Date());

DBEVDocument dbevDocument = new DBEVHelper().newDBEVDocument();

// Set mandatory constraints
dbevDocument.setTargetId(patientId);
dbevDocument.setTargetTypeCode("Patient");
dbevDocument.setSendingSystemCode("DBEV");
dbevDocument.setOwnerId(patientId);
dbevDocument.setContents(new VAOInterfaceBase[]{existingAct});
dbevDocument = mDBEVService.createDBEVs(
new DBEVDocument[]{dbevDocument})[0];

// Submit for approval
mDBEVService.submitDBEVs(new String[]{dbevDocument.getId()});
// Approve ( commit );
mDBEVService.approveDBEVs(new String[]{dbevDocument.getId()});
```

4.9.3 Implementing User Catalogs

A user catalog is an optional container for templates and organizational folders. There are two types of user catalogs that can be used in HTB: (i) User Catalogs, and (ii) Formulary User Catalogs.

User catalogs support the creation and maintenance of templates that facilitate creation of specific clinical acts such as medication orders. Formulary User Catalogs support creation and maintenance of templates that assist in grouping and organizing a clinical formulary. For example, a formulary can be created for orderable drugs in an inpatient pharmacy, or for medications that are covered by a health insurance plan.

User catalogs are created and deleted by the User Catalog Owner (defined in Section 4.1.3), and used by user catalog owners and user catalog viewers. Contents of a user catalog are defined by this user catalog owner.

User catalogs and formulary user catalogs can contain the following items:

- Clinical Act Template Links: These link a clinical act template to a user catalog
- Folders used to organize access to Clinical Act Template Links

Individual users can construct personal templates using individual user catalogs. User catalogs do not store individual patient information.

There are two phases in the implementation of user catalogs: (i) Creating the user catalog container, and (ii) creating the user catalog content. The latter is a continuous process in which user catalog items are created, modified, and deleted. User catalog constructs are summarized in Table 4–90:

Table 4–90 User Catalogs

Construct	Definition
User Catalog	The `UserCatalog` object represents a single catalog of `UserCatalogItem` value objects.
User Catalog Folder	A `UserCatalogFolder` is a `UserCatalogItem` that represents a folder within the User Catalog. A `UserCatalogFolder` can contain other `UserCatalogItem` objects, including other `UserCatalogFolder` objects. Named constructs that contain flow sheet definitions or nested folders of flow sheet definitions. Used to organize content in the User Catalog.
Clinical Act Template Link	A Clinical Act Template Link is a `UserCatalogItem`. Clinical templates are linked to a user catalog via a clinical act template link. Clinical templates facilitate the use of logical combinations of clinical acts that can be used generically. To create a clinical act template, a clinical act that is not specific to any patient information or staff participations is defined. A user catalog template and a clinical act are linked through reference to the clinical act ID.

User catalogs support the following business services:

- Creation of clinical act templates that let organizations save default data, to standardize order and data entry practices.
- Creation of user catalog folders, to support query and display of clinical template collections.

Reference

Oracle Javadoc for HTB

Table 4–91 lists the principal user catalog service and methods:

Table 4–91 Service and Methods: Implementing User Catalogs and Flow Sheets

Level	Detail
Package	`oracle.apps.ctb.clinical.usercatalog`
Class	`UserCatalogService`
Methods	■ `createUserCatalogItems` ■ `createUserCatalogs` ■ `findTemplates` ■ `findUserCatalogItems` ■ `findUserCatalogs` ■ `updateUserCatalogItems` ■ `updateUserCatalogs`

Prerequisites

- Implementing Organizations
- Implementing Staff Management
- Implementing Workgroups if viewers are required
- Implementing Master Catalogs

See Also:

- Section 4.3, Implementing Organizations
- Section 4.8.4, Implementing Staff Management
- Section 4.8.8, Implementing Workgroups
- Section 4.9.1, Implementing Master Catalogs

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

- Use the responsibility you defined in Section 4.1.4.
- User Catalog Owner - staff user

Navigation

This is an API-based implementation procedure.

See Appendix A, HTB Session Service

4.9.3.1 Implementing User Catalog Containers

Steps

Perform the following steps to create a User Catalog that contains templates:

1. Use the `createUserCatalogs` method to create the user catalog ID.
2. Use the `CreateUserCatalogs` method to name the user catalog.
3. Use the `CreateUserCatalogs` method to assign owners from the list of staff users.

Code Sample: Creating a User Catalog

Example 4–12 illustrates how to create two new `UserCatalog` objects— (i) one of type `UserCatalog`, and (ii) the other of type Formulary. Call `UserCatalogHelper` to set the type.

Example 4–12 Code Sample: Creating a User Catalog

```
UserCatalogHelper ucHelper = new UserCatalogHelper();
UserCatalog userCatalog1 = ucHelper.newUserCatalog();
userCatalog1.setUserCatalogName("Test UserCatalog 1");

UserCatalog userCatalog2 = ucHelper.newUserCatalogFormularies();
userCatalog2.setInsurancePlanId("1");
userCatalog2.setUserCatalogName("Test UserCatalog 2");

UserCatalog[] createdUserCatalogs = userCatalogService.createUserCatalogs(new
```

```
UserCatalog[] {userCatalog1, userCatalog2});
```

Updating a User Catalog

4. Use the `findUserCatalogs` and `updateUserCatalogs` methods to locate and update the user catalog.
5. Use the `updateUserCatalog` method to perform tasks such as updating effective time (for performing a soft delete of the user catalog) by setting an effective end time.

4.9.3.2 Implementing User Catalog Contents

User catalog owners can specify the contents of a user catalog, including catalog folders and templates (clinical act template link). The user catalog folders are used for organizing logical access to templates. These folders are not required in cases where a list of templates suffices. However, a hierarchy of user catalog folders can be constructed if the structure is complex.

Implementing Folders

Steps

1. Use the `getUserCatalog` method to access the user catalog.
2. Use the `createUserCatalogItems` method to create a user catalog folder.

Example 4–13 Code Sample: Creating a User Catalog Folder

```
UserCatalogHelper ucHelper = new UserCatalogHelper();
UserCatalogFolder folder = ucHelper.newUserCatalogFolder();
folder.setUserCatalogFolderName("Test UserCatalogFolder");
folder.setUserCatalogId("1");  // the UserCatalog this belongs to

UserCatalogItem[] createdItems = userCatalogService.createUserCatalogItems(new
UserCatalogItem[] {folder});
```

3. To create a user catalog sub-folder, use the `createUserCatalogItems` method. Select the existing user catalog folder and create and name the sub-user catalog folders.
4. To create a clinical act and a clinical act template link, see Example 4–14 and Example 4–15 for code samples.

Updating a User Catalog Folder

To update a user catalog item, you can optionally use the `findUserCatalogItems` method with the `findTemplates` method. Select the target catalog item and use the `updateUserCatalogItems` method to update it.

Implementing Clinical Act Templates for a User Catalog

Clinical act templates are linked to a user catalog by clinical act template links. Clinical act templates are clinical acts, created by Clinical Act Service methods. User catalog owners can specify the clinical act templates that are associated with a user catalog.

The steps for creating templates are similar to creating clinical act instances with the following exceptions: When creating a clinical act template, (i) you will not associate clinical act templates in act participations, and (ii) you will not assign templates to specific patients.

Steps

1. Determine the clinical domain for which clinical act templates are required.
2. To create a clinical act template instance, you must determine what clinical activity is to be represented and the materials required for the activity (example: medication order and some material substance), where the activity takes place (example: organization or facility where the care is being provided), and when the activity will take place.
3. Select the act definition from the master catalog that applies to the type of clinical act that you want to create.

   See Also: Example 4–9, Code Sample: Finding an Act Definition,

4. Use Clinical Act Service methods to create a clinical act template instance:
   - Use the `createTemplates` method to create a substance administration act template.
   - Set the values of the clinical act attributes in accordance with the information captured in step 2.

Example 4–14 Code Sample: Creating a Medication Template

```
String actDefinitionId = "1";
// for example, this is Class = "SBADM", Mood = "RQO", Category = "CLIN_DRUG"
```

```
ActHelper helper = new ActHelper();
SubstanceAdministrationAct medicationTemplate =
helper.newSubstanceAdministrationAct();

// note, templates do not have a status, or refer to a patient, encounter, or
// medical record, and cannot have any ActParticipations
medicationTemplate.setActDefinitionId(actDefinitionId);
medicationTemplate.setUpdateMode(ClinicalAct.UPDTMD_NEW);
PhysicalQuantity physicalQuantity = helper.newPhysicalQuantity();
physicalQuantity.setQuantity("500");
physicalQuantity.setUnit("mg");
medicationTemplate.setDoseQuantity(physicalQuantity);

ClinicalAct[] createdTemplates = clinicalActService.createTemplates(new
ClinicalAct[] {medicationTemplate});
```

5. Link the clinical act template to the user catalog by creating a clinical act template link:

Example 4–15 Code Sample: Creating a Clinical Act Template Link

```
UserCatalogHelper ucHelper = new UserCatalogHelper();
ClinicalActTemplateLink templateLink = ucHelper.newClinicalActTemplateLink();
templateLink.setTemplateClinicalActId("100000");
templateLink.setUserCatalogId("1");  // the UserCatalog this belongs to

UserCatalogItem[] createdItems = userCatalogService.createUserCatalogItems(new
UserCatalogItem[] {templateLink});
```

6. Finding a Clinical Act Template Instance: Searching for a clinical act template is performed to (i) update a template, and (ii) to find and use a template to place an order. To find the clinical act template instance, select the user catalog associated with the clinical act template and use the `FindTemplates` method to perform the search.

   The code sample in Example 4–16 searches for Substance Administration Request templates within a specified user catalog:

Example 4–16 Code Sample: Retrieving a Medication Template in a User Catalog

```
ActHelper helper = new ActHelper();
ClinicalTemplateQueryCriteria templateQC =
helper.newClinicalTemplateQueryCriteria();
SearchTerm stClass =
templateQC.createSearchTerm(ClinicalTemplateQueryCriteria.CLASS_CODE, "SBADM");
```

```
SearchTerm stMood =
templateQC.createSearchTerm(ClinicalTemplateQueryCriteria.MOOD_CODE, "RQO");
templateQC.setRootSearchTerm( templateQC.and(stClass, stMood) );

// we want to search for our templates in a particular UserCatalog
UserCatalogHelper ucHelper = new UserCatalogHelper();
UserCatalogItemDependentQueryCriteria ucItemQC =
ucHelper.newUserCatalogItemDependentQueryCriteria();
SearchTerm stUserCatalogId =
ucItemQC.createSearchTerm(UserCatalogItemDependentQueryCriteria.USER_CATALOG_ID,
"1");
ucItemQC.setRootSearchTerm(stUserCatalogId);

ClinicalAct[] foundTemplates = userCatalogService.findTemplates(templateQC,
ucItemQC);
```

4.9.4 Implementing Clinical Applications

HTB Clinical Business Services provides a unified data model based on the HL7 Version 3 **Reference Information Model** (RIM). Tables 4–92 and 4–93 summarize the development process:

Table 4–92 Building Clinical Applications: Overview [1]

Global Prerequisites	Maintain User Catalog	Medication Order Entry	Results Reviewing	Medication Administration Record
Administrative Business Services	Yes	Yes	Yes	Yes
Enterprise Terminology Services (ETS)	Yes	Yes	Yes	Yes
Master Catalog	Yes	Yes	Yes	Yes

1 *Yes* indicates use of HTB components recommended.

Table 4–93 Building Clinical Applications: Application Prerequisites, Functionality[1]

Application Functionality	Maintain User Catalog	Medication Order Entry	Results Reviewing	Medication Administration Record
Patient Demography Display	NA	Yes	Yes	Yes

Table 4–93 (Cont.) Building Clinical Applications: Application Prerequisites,

Encounter Display	NA	Yes	Yes	Yes
Allergy List	NA	Yes	Yes	Yes
User Catalog - Templates	Yes	Yes	NA	NA
Knowledge Information Document	NA	Yes	NA	Yes
Application DBEV	NA	Yes	NA	Yes

[1] *Yes* indicate use of HTB components recommended.

The following section provides code samples that illustrate some of the application functionality provided by Clinical Business Services. These samples can be displayed in an instance of one patient to provide patient context to the user of the clinical application.

4.9.4.1 Patient Demographics Display

Displaying patient demographic information could include patient name, date of birth, and gender. The code sample in Example 4–17 assumes that the patient record has been retrieved:

Example 4–17 Code Sample: Demographic Display

```
PersonName patientNames = patient.getPerson().getNames();
String patientFirstName = patientNames[0].getFirstName();
String patientMiddleName = patientNames[0].getMiddleName();
String patientLastName = patientNames[0].getLastName();
Date patientDOB = patient.getPerson().getDateOfBirth();
String patientGender = patient.getPerson().getGender();
MedicalRecord[] medicalRecords = patient.getMedicalRecords();
for (int i=0; medicalRecords != null && i<medicalRecords.length; i++)
{
    String medicalRecordNumber = medicalRecords[i].getMedicalRecordNumber();
}
```

4.9.4.2 Encounter Display

The code sample in Example 4–18 assumes that displaying encounter information could include the user (Staff) name, encounter ID, and care site, and that a current encounter record has been retrieved:

Example 4–18 Code Sample: Encounter Display

```
String encounterId = encounter.getIdentifier();
EncounterCareSite[] careSites = encounter.getCareSites();
for (int i=0; careSites != null && i<careSites.length; i++)
{
    // note: careSites have both a care site id and a practice setting id
    String careSiteId = careSites[i].getLocation().getCareSiteIdentifier();
    String practiceSettingId =
        careSites[i].getLocation.getPracticeSettingIdentifier();
    // we can now display these if we like, or get more detail
}

EncounterPracticeSetting[] practiceSettings = encounter.getPracticeSettings();
for (int i=0; practiceSettings != null && i<practiceSettings.length; i++)
{
    String practiceSettingId =
        practiceSettings[i].getLocation.getPracticeSettingIdentifier();
    // we can now display these if we like, or get more detail
}

// we want the admitting physician - this will be an EncounterStaffParticipant
// with a Role of "ADM"
EncounterStaffParticipant[] staffParticipants =
    encounter.getStaffParticipants();
for (int i=0; staffParticipants != null && i<staffParticipants; i++)
{
    if (staffParticipants[i].getParticipatingRole().equals("ADM"))
    {
        String admittingPhysicianId =
            staffParticipants[i].getStaffMemberIdentifier();
        // and now we can display this if we like
    }
}
```

4.9.4.3 Allergy List

Displaying an allergy list could include a list of allergies that the patient is allergic to, such as penicillin and peanuts.

Code Sample

See Example 4–20 for a code sample.

4.9.4.4 User Catalog Display

The code sample in Example 4–19 assumes that elements that could be displayed include user catalogs that the user (Staff) has access to and that the user (Staff) is known:

Example 4–19 Code Sample: List User Catalogs where the logged-in user is a member of an associated Workgroup

```
// assuming we know the staff member's position from somewhere
StaffPosition currentStaffPosition = ...;
String staffPositionId = currentStaffPosition.getId();

// we want to find all Workgroups that the user is associated with
WorkgroupMembership[] staffWorkgroups =
workgroupService.findWorkgroupMemberships(null, staffPositionId, null, null);

// we want to find all UserCatalogs for ALL the user's Workgroups
// To do this, we need to loop though the workgroups and set each id
// on an individual UserCatalogWorkgroupDependentQueryCriteria and 'or' them
// together.

UserCatalogHelper userCatalogHelper = new UserCatalogHelper();
UserCatalogQueryCriteria userCatalogQC =
userCatalogHelper.newUserCatalogQueryCriteria();
UserCatalogWorkgroupDependentQueryCriteria userCatalogWorkgroupQC =
userCatalogHelper.newUserCatalogWorkgroupDependentQueryCriteria();

SearchTerm workgroupIdsST = null;
for (int i=0; i< staffWorkgroups.length; i++)
{
// create new temporary SearchTerm
SearchTerm currentIdST = userCatalogWorkgroupQC.createSearchTerm(
UserCatalogWorkgroupDependentQueryCriteria.WORK_GROUP_ID,
staffWorkgroups[i].getWorkgroupId());
// add to complete workgroup SearchTerm
workgroupIdsST = (workgroupIdsST == null) ? currentIdST                    :
```

```
userCatalogWorkgroupQC.or(workgroupIdsST, currentIdST);
}

userCatalogWorkgroupQC.setRootSearchTerm(workgroupIdsST);
SearchTerm workgroupST =
userCatalogQC.createUserCatalogWorkgroupDependentQueryCriteria(
userCatalogWorkgroupQC);

userCatalogQC.setRootSearchTerm(workgroupST);

UserCatalog[] foundUserCatalogs =
userCatalogService.findUserCatalogs(userCatalogQC);

for (int i=0; foundUserCatalogs != null && i<foundUserCatalogs.length; i++)
{
    String userCatalogName = foundUserCatalogs[i].getUserCatalogName();
   // can now display this...
}
```

4.9.4.5 Knowledge Information Document

Elements that could be displayed from a knowledge information document may include active medication orders, medication histories, allergies, weight, height, body surface index, and body surface area.

Code Sample

See Example 4–22 for a code sample.

4.9.4.6 Application DBEV

This functionality lets users enter and submit clinical data into the HTB Repository. Examples include populating a medication order template for entry and submission for a specific patient.

Code Sample

See Example 4–10 for a code sample.

4.9.5 Implementing a Medication Order Entry System

Clinical Business Services assists in the management of medication orders, events, and medication supply events. This section describes an illustrative Medication Order Entry System application that can be developed using this functionality.

Reference

Oracle Javadoc for HTB

Table 4–94 lists the principal methods used to implement Clinical Acts:

Table 4–94 Service and Methods: Implementing Clinical Acts

Level	Detail
Package	`oracle.apps.ctb.clinical.act`
Class	`ClinicalAct`
Methods	▪ `createActInstance` ▪ `createTemplates` ▪ `findAlerts` ▪ `findClinicalActs` ▪ `findClinicalActXRefByII` ▪ `findMaterialXRefByII` ▪ `findTemplates` ▪ `getActInstanceDetail` ▪ `getActInstanceHistory` ▪ `getClinicalActByExternalId` ▪ `getKnowledgeDocumentWithDBEV` ▪ `getMaterialByExternalId` ▪ `getMessageContext` ▪ `updateActInstance` ▪ `updateClinicalActDocuments` ▪ `updateTemplates`

Prerequisites

- Implementing Security Services
- Implementing Person Services
- Implementing Organizations
- Implementing Enterprise Terminology Services
- Implementing Audit Services
- Implementing Administrative Business Services

- Implementing Master Catalogs:
- Implementing User Catalogs

  The User Catalog is the mechanism used to organize and create templates (clinical acts with default data) that can be represented in a medication order entry application.

  See Also:

  - Section 4.1, Implementing Security Services
  - Section 4.2, Implementing Person Services
  - Section 4.3, Implementing Organizations
  - Section 4.5, Implementing Enterprise Terminology Services
  - Section 4.6, Implementing Audit Services
  - Section 4.8, Implementing Administrative Business Services
  - Section 4.9.1, Implementing Master Catalogs
  - Section 4.9.3, Implementing User Catalogs

Implementing a Medication Order Entry System

The following sections describe the assumptions, functionality, and elements of the implementation process:

- Section 4.9.5.1, Medication Order Entry System Overview
- Section 4.9.5.2, Medication Order Entry System Functionality
- Section 4.9.5.3, Medication Order Entry System Implementation Elements

4.9.5.1 Medication Order Entry System Overview

A medication order entry system is used by **clinician**s in various healthcare settings to generate medication orders for direct transmission to **inpatient** and **outpatient** pharmacies.

A medication order entry system has patient and medication context. Applications features that display a patient search feature or a patient list feature provide users with the patient context for creating and managing medication orders.

Medication order entry is typically based on searching for a particular medication that the physician is ordering and then entering the attribute data. To streamline

this process, HTB provides a mechanism to create medication templates that can be saved, searched on and used in the order entry process.

Templates of medication orders are most effectively created and managed through the use of the HTB User Catalog. The user catalog is a holding place for all the medication templates that a user or system administrator creates. Each medication item (template) in the user catalog references an Act Definition in the master catalog. Each medication template is a **clinical act** in *order* or *event* **mood**, and is stored in the HTB Repository as a clinical act—with the exception that the act is not associated with a patient.

A medication order entry system can optionally utilize drug knowledge integration tools that provide mechanisms to pass the medication order and a subset of pertinent patient specific data (such as gender, date of birth or age, active medication orders, active allergies, and lab results) to a third party knowledge engine.

The third party knowledge base processes the data it receives and returns to the application alerts, and reference data. Knowledge integration can be implemented either interactively during the order entry process, or on demand when a user requests that a drug check be performed. These features are determined by the application. HTB supports this process by retrieving all active medication orders, active allergies, active medication history events, and any in process medication orders from the HTB repository. This collection of data is called a Knowledge Integration Document within HTB.

4.9.5.2 Medication Order Entry System Functionality

- The user interface is the responsibility of the application. Issues to consider:
  - Screen density.
  - Variety of data displayed on a single page (such as person and patient data, encounter data, allergy list, navigation, and patient context).
  - Order entry functions, such as searching or scrolling through a list of available medications.
- Patient selection is performed by searching for a patient and selecting the target patient from the search results, or by displaying a list of patients and selecting a patient from the patient list.
- The application will let you create medication orders.
- Searching for a medication to order. This could be performed by searching for a template in one of the user catalogs.

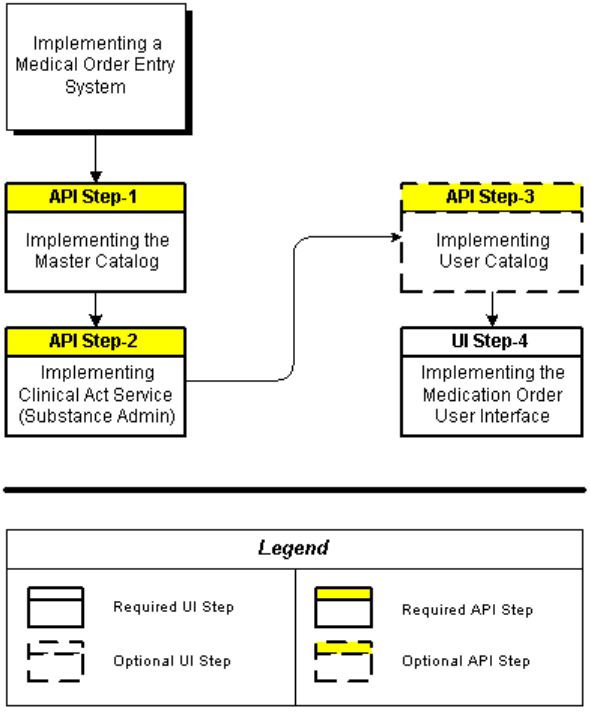
- The application will let you select the drug to order.
- Entering attribute data for the order such as dose, priority, form, and route.
- If the selected drug is based on a template, most of the attributes can have default data and minimal data entry is required to complete the order entry process.
- Optional implementation of drug checks such as drug-to-drug interactions is the responsibility of the application.
- The application will let you submit or create an order.
- Creating a complex medication order is similar to creating any medication order, but also includes creating the relationships between several different medications. An example of a complex medication order is a Total Parental Nutrition (TPN) order. This order calls for a base and several additives in component act relationships to the parent act TPN.
- Finding medication order history, querying for active orders or for orders in any state. Typically, these are lists of orders for the respective patient.
- Updating medication orders: As orders are cancelled, revised, or marked as entered in error, the state changes are tracked in HTB and the status of the order is always known.

Note: In order for the medication order entry system to show that an order has been carried out or administered, a fulfillment action relationship must be created between the substance administration event and the original substance administration request for order.

4.9.5.3 Medication Order Entry System Implementation Elements

This section describes the implementation process for an illustrative Medication Order Entry System. Figure 4–21 provides an overview of the process:

Figure 4–21 Implementing a Medication Order Entry System



The following sections describe the implementation elements of the sample application:

- Creating a Master Catalog
- Creating a User Catalog
- Creating Medication Order Templates
- Finding Medication Order Templates
- Updating Medication Order Templates
- Creating Substance Administration Acts
- Revising Substance Administration Acts
- Finding Clinical Acts

Creating a Master Catalog

See the following references for information about creating a master catalog:

- Section 4.9.1, Implementing Master Catalogs
- Table 4–86, Service and Methods: Implementing the Master Catalog

Creating a User Catalog

See the following references for information about creating user catalogs:

- Section 4.9.3, Implementing User Catalogs
- Table 4–91, Service and Methods: Implementing User Catalogs and Flow Sheets

Creating Medication Order Templates

Assume that a User Catalog has been created to organize medication order templates.

See the following references for information about creating medication order templates:

- Section 4.9.2, Implementing Clinical Act Service
- Example 4–14, Code Sample: Creating a Medication Template
- Table 4–89, Service and Methods: Implementing Clinical Acts

Finding Medication Order Templates

Steps

1. Specify the User Catalog Identifier and use the `findUserCatalog` method to find and select the user catalog the template is associated with.
2. Use the `findTemplates` method to retrieve a list of clinical templates and their related templates that match the search criteria, which are defined by the `UserCatalogQueryCriteria` value object.

   See Also: Example 4–16, Code Sample: Retrieving a Medication Template in a User Catalog

Updating Medication Order Templates

Steps

1. Use the methods `userCatalogQueryCriteria` and `findUserCatalogs` to select the user catalog to be searched, by specifying the user catalog identifier.
2. Use the `findUserCatalogItem` method to retrieve a list of clinical templates and their related templates that match the search criteria, which is defined by the `UserCatalogQueryCriteria` value object (use the method `findUserCatalogItems`).
3. Select the template to be updated and use the `updateUserCatalogItems` method to update the selected template.

Creating Substance Administration Acts

Steps

1. A Substance Administration Act is a subclass of clinical act. Use Clinical Act Service methods (Table 4–94) to create, update, delete, and find act instances. The interface `SubstanceAdministrationAct` (`oracle.apps.ctb.clinical.act`) contains methods specific to the subclass.
2. Determine the type of clinical data that will be created and viewed within the order entry system. In a medication order entry system, a developer would typically use substance administration request for order, substance administration event, observation event, and supply event. Each of these areas requires one or more act definitions.
3. From the master catalog, use the `findActDefinitions` method to select the act definition that applies to the type of clinical act that you intend to create. For example, you might select one of the following `SubstanceAdministrationAct` definitions:
   - Class: *Substance Administration*
   - Mood: *Request for Order*
   - Category Code: *Clinical Drug*
   - Detail Id: *Furosemide*
   - Detail Type Code: *ID*
   - Effective Start Time: *[yyyymmdd]*

- Organization: *[nnn]*

4. Use the `createActInstance` method to create a `SubstanceAdministrationAct`.
5. Assign values to the `SubstanceAdministrationAct` attributes accordingly.

See Also: Example 4–10, Code Sample: Creating a new Medication Order using DBEV

Revising Substance Administration Acts

Steps

1. Use the methods `ClinicalActQueryCriteria`, `FindClinicalActs`, and `GetInstanceDetail` to find the `SubstanceAdministrationAct` of interest using specified query parameters and display act details; select the `SubstanceAdministrationAct` to update.
2. Use the Clinical Act Service method `updateActInstance` to update the selected Substance Administration Act**.**

See Also: Example 4–11, Code Sample: Update Clinical Acts with DBEV

Finding Clinical Acts

Finding Active Allergies

Steps

Lists of acts: Use the methods `findClinicalAct` **and** `ClinicalActQueryCriteria` **to find** all active allergies for the selected patient.

Sample Query Parameters:

- Patient ID: *John Smith (123123)*
- Class: *Observation (OBS)*
- Mood: *Event (EVN)*
- Category: *Allergy (ALL)*
- Status: *Active*

Code Sample: List of all Active Intolerances

Assumption: We can perform this search in one of two ways:

- Use the `findClinicalAct` and `ClinicalActQueryCriteria` methods.
- Retrieve a knowledge document for the patient, and get the Act details with the `getActiveAllergies` method (Example 4–22).

Display details are retrieved from the returned Acts.

Example 4–20 Code Sample: Using `ClinicalActService`

```
ActHelper actHelper = new ActHelper();
int retrievalDepth = 10;  // Arbitrary depth
String patientId = "12345";  // the patient we wish to query on
boolean forceFollowInseparable = false;
// types of relationships we wish to traverse in the query
java.lang.String[] traversableRelationships = null;
// we're not going to set the child criteria, but we do need it for the query
ClinicalActQueryCriteria childQC = actHelper.newClinicalActQueryCriteria();

ClinicalActQueryCriteria qc = actHelper.newClinicalActQueryCriteria();

SearchTerm stClass = qc.createSearchTerm(qc.CLASS_CODE, "OBS");
SearchTerm stMood = qc.createSearchTerm(qc.MOOD_CODE, "EVN");
SearchTerm stCategory = qc.createSearchTerm(qc.CATEGORY_CODE, "ALL");
SearchTerm stStatus = qc.createSearchTerm(qc.STATUS_CODE , "active");
SearchTerm stPatient = qc.createSearchTerm(qc.PATIENT_ID, patientId);

// combine the Search Terms and set the root Search Term of the Query
SearchTerm stClassAndMood = qc.and(stClass, stMood);
SearchTerm stClassMoodCategory = qc.and(stClassAndMood, stCategory);
SearchTerm stPatientAndStatus = qc.and(stPatient, stStatus);

qc.setRootSearchTerm( qc.and(stClassMoodCategory, stPatientAndStatus) );

ObservationAct[] acts = (ObservationAct[])clinicalActService.findClinicalActs(
        qc,
        childQC,
        retrievalDepth,
        traversableRelationships,
        forceFollowInseparable);

// to get the details of the allergen, we can do the following:
for (int i=0, i<acts.length; i++)
{
```

```
    String allergenName = acts[i].getObservationEtsValueDescription();
    Date allergyStarted = acts[i].getEffectiveStartTime();
    Date allergyEnded = acts[i].getEffectiveEndTime();
    // these can then be displayed, etc
}
```

Finding Active Substance Administration Orders

Lists of acts: Use the `findClinicalAct` and `ClinicalActQueryCriteria` methods to find all active medication orders for the selected patient.

Sample Query Parameters:

- Patient ID: *John Smith 123123*
- Class: *Substance Administration (SBADM)*
- Mood: *Request for Order (RQO)*
- Category: *(Any)*
- Status: *Active*
- Detail Id: *(Any)*

Example 4–21 Code Sample: Finding all Active Substance Administration Orders for a Specified Encounter

```
String encounterId = "1";  // the encounter id we're looking for
ActHelper actHelper = new ActHelper();
ClinicalActQueryCriteria qc = actHelper.newClinicalActQueryCriteria();
SearchTerm stClass = qc.createSearchTerm(ClinicalActQueryCriteria.CLASS_CODE,
"SBADM");
SearchTerm stMood = qc.createSearchTerm(ClinicalActQueryCriteria.MOOD_CODE,
"RQO");
SearchTerm stStatus = qc.createSearchTerm(ClinicalActQueryCriteria.STATUS_CODE,
"active");
SearchTerm stEncounter = qc.createSearchTerm(ClinicalActQueryCriteria.ENCOUNTER_
ID, encounterId);
SearchTerm stClassMood = qc.and(stClass, stMood);
SearchTerm stStatusEnc = qc.and(stStatus, stEncounter);
qc.setRootSearchTerm( qc.and(stClassMood, stStatusEnc) );

ClinicalAct[] foundActs = clinicalActService.findClinicalActs(qc);
```

Note: *The code sample for displaying clinical act details is similar to Example 4–20.*

Finding a Knowledge Integration Document

Use the `GetKnowledgeDocumentwithDBEV` method to query for a knowledge integration document.

Assumption: If there is any DBEV information that needs to be considered, we would include the identifiers in the `dbevIds` array. For this example we have not specified any of these, but it would not change the code for retrieving the information.

Example 4–22 Code Sample: Retrieving Knowledge Document information

```
String[] dbevIds = {};  // ids of work-in-progress information
String patientId = "12345"; // patient to query on

KnowledgeDocument knowledgeDoc =
clinicalActService.getKnowledgeDocumentWithDBEV(patientId, dbevIds);

ObservationAct[] activeAllergies = knowledgeDoc.getActiveAllergies();
SubstanceAdministrationAct[] activeMedHistories =
knowledgeDoc.getActiveMedicationHistory();
SubstanceAdministrationAct[] activeMedOrders =
knowledgeDoc.getActiveMedicationOrders();

PhysicalQuantity currentWeight = (PhysicalQuantity)knowledgeDoc.getWeight();
PhysicalQuantity currentHeight = (PhysicalQuantity)knowledgeDoc.getHeight();
PhysicalQuantity currentBMI = (PhysicalQuantity)knowledgeDoc.getBodyMassIndex();
PhysicalQuantity currentBSA =
(PhysicalQuantity)knowledgeDoc.getBodySurfaceArea();

// to get the details of the allergen, we can do the following:
for (int i=0; i<activeAllergies.length; i++)
{
    String allergenName =
activeAllergies[i].getObservationEtsValueDescription();
    Date allergyStarted = activeAllergies[i].getEffectiveStartTime();
    Date allergyEnded = activeAllergies[i].getEffectiveEndTime();
    // these can then be displayed, etc
}
```

4.9.6 Implementing a Results Reviewing System

HTB Clinical APIs assist in the retrieval and review of observation events such as laboratory results. This section describes an illustrative application that can be developed using Clinical Business Services.

Prerequisites

- Implementing Security Services
- Implementing Person Services
- Implementing Organizations
- Implementing Administrative Business Services
- Implementing Clinical Business Services
- Implementing Master Catalogs

See Also:

- Section 4.1, Implementing Security Services
- Section 4.2, Implementing Person Services
- Section 4.3, Implementing Organizations
- Section 4.8, Implementing Administrative Business Services
- Section 4.9, Implementing Clinical Business Services
- Section 4.9.1, Implementing Master Catalogs

The following sections describe assumptions, functionality, and the implementation elements for the sample application:

- Section 4.9.6.1, Results Reviewing System Overview
- Section 4.9.6.2, Results Reviewing System Functionality
- Section 4.9.6.3, Results Reviewing System Implementation Elements

4.9.6.1 Results Reviewing System Overview

A laboratory resulting system is used by clinicians in various healthcare settings to retrieve and review laboratory results for selected patients.

A laboratory resulting system has patient and laboratory result context. Laboratory result review is often based on a clinician searching for a patient and reviewing labs for that patient. Applications based on HTB therefore would feature the ability to search for patients or to have a patient list feature that would provide users the patient context for retrieving and managing laboratory results.

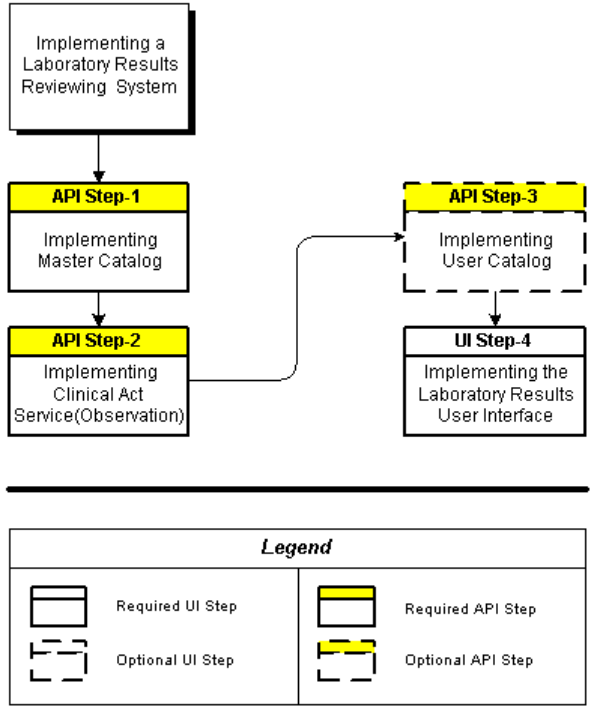
4.9.6.2 Results Reviewing System Functionality

- The user interface is the responsibility of the application. Issues to consider:
  - Screen density.
  - Variety of data displayed on a single page (person and patient data, encounter data, allergy list, navigation, patient context, and laboratory result viewing functions such as searching or scrolling through a list of laboratory results).
  - Order entry functions, such as searching or scrolling through a list of available medications.
- Patient selection is performed by searching for the patient and selecting the patient from the search results, or by displaying a list of patients and selecting a patient from the patient list.
- Searching and Reviewing Laboratory Results:

4.9.6.3 Results Reviewing System Implementation Elements

This section describes the implementation process for an illustrative Results Reviewing System. Figure 4–22 provides an overview of the process:

Figure 4–22 Implementing a Results Reviewing System



The following sections describe the implementation elements for the sample application:

- Creating a Master Catalog
- Working with Clinical Acts

Creating a Master Catalog

See the following references for information about creating a master catalog:

- Section 4.9.1, Implementing Master Catalogs
- Table 4–86, Service and Methods: Implementing the Master Catalog

Working with Clinical Acts

To find Clinical Acts, in this case laboratory results for a particular patient, use the `findClinicalActs` and `ClinicalActQueryCriteria` methods to find all completed laboratory events for a selected patient.

Sample Query Parameters:

- Patient ID: *John Smith (123123)*
- Class: *Observation (OBS)*
- Mood: *Event (EVN)*
- Category: *Laboratory (LAB)*
- Status: *Completed*

See Also: Example 4–21, Code Sample: Finding all Active Substance Administration Orders for a Specified Encounter

4.10 Implementing Cross-Referencing

Cross-referencing refers to the association of related objects across systems, so that the same object found in multiple repositories can be consistently identified. Cross-referencing services let you map externally assigned identifiers of an object to its internal (HTB-assigned) identifier. The implementation procedures in this section describe the process of creating cross-references between external and internal identifiers using the HTB application programming interface.

Role of Cross-References in Messaging

The HL7 standard requires use of instance identifiers to identify objects in messages, such as persons, patients, or clinical acts. Each instance identifier is a combination of a root and an extension. The root is an object identifier (OID) that guarantees the uniqueness of the extension. For organization objects, the root is the complete instance identifier. The extension is the unique identifier for the object within the scope of the root.

When Inbound Messaging Services (IMP) receives a message from an external system, each identifiable object in the message is associated with one or more external instance identifiers. IMP first attempts to resolve these identifiers to an existing object in HTB. If they cannot be resolved and IMP configuration rules permit creation of such an object, HTB creates the object and assigns an internal identifier to it. It also cross-references the external identifiers from the message to this internal identifier. Future references to this object in inbound messages are resolved to the HTB object through the cross-references. When Outbound Messaging Services (OMP) sends messages about this object to an external system, it includes the internal identifier as well as external identifiers from all of its cross-references as instance identifiers in the message.

IMP has a configuration that identifies valid external and internal roots for each object type. Each external root used in this configuration must be registered with HTB using the procedure described in Section 4.10.1.

For HTB objects that can be referenced in messages but cannot be created by IMP, cross-references must be created using the procedure described in Section 4.10.2. Typically this includes organizations and care sites because they cannot be created by messages. Other object cross-references may have to be set up in this manner if IMP's trigger and side effect configuration prohibit their creation.

Notes:

- A cross-reference is composed of an external root and extension, an internal identifier and an interval of time during which the cross-reference is considered valid.
- An external identifier can be associated with multiple internal identifiers, but the valid times cannot overlap; an internal identifier can be associated with multiple external identifiers.
- Only the valid start time and valid end time can be updated in an existing cross-reference.
- The valid start time is *inclusive*; the valid end time is *exclusive*. A null value for valid end time indicates that the instance identifier cross-reference is valid indefinitely. A null valid start time indicates that it is indeterminate.

Reference

Oracle Javadoc for HTB

Table 4–95 lists the principal cross-referencing services and methods:

Table 4–95 Service and Methods: Cross-Referencing

Level	Detail
Package	`Several...search Javadoc for method name`
Class	`Several...search Javadoc for method name`

Table 4–95 (Cont.) Service and Methods: Cross-Referencing

Level	Detail
Methods	■ CareSite.setCareSiteXRefs ■ ClinicalAct.setClinicalActXRefs ■ CorrelatePersonService.createCorrelateData ■ Encounter.setEncounterCrossReferences ■ findCareSiteXRefByII ■ findClinicalActXRefByII ■ findEncounterXRefsByInstanceIdentifier ■ findMaterialXRefByII ■ findMedicalRecordXRefByII ■ findPersonByExternalId ■ findStaffMemberXRefByII ■ findStaffPositionXRefByII ■ getCareSiteByExternalId ■ getClinicalActByExternalId ■ getEncounterByExternalIdentifier ■ getMaterialByExternalId ■ getMedicalRecordByExternalId ■ getOrgUnitByExternalId ■ getStaffMemberByExternalId ■ getStaffPositionByExternalId ■ getOrgUnitByExternalId ■ Material.setMaterialXRefs ■ MedicalRecord.setMedicalRecordXRefs ■ OrgUnit.setExternalOrgId ■ StaffMember.setXRefs ■ StaffPosition.setXRefs

See Also:

- Section 4.2, Implementing Person Services
- Section 4.2.5, Implementing Person Management
- Section 4.2.6, Implementing Correlation

...for more information about person cross-referencing.

Prerequisites

None.

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

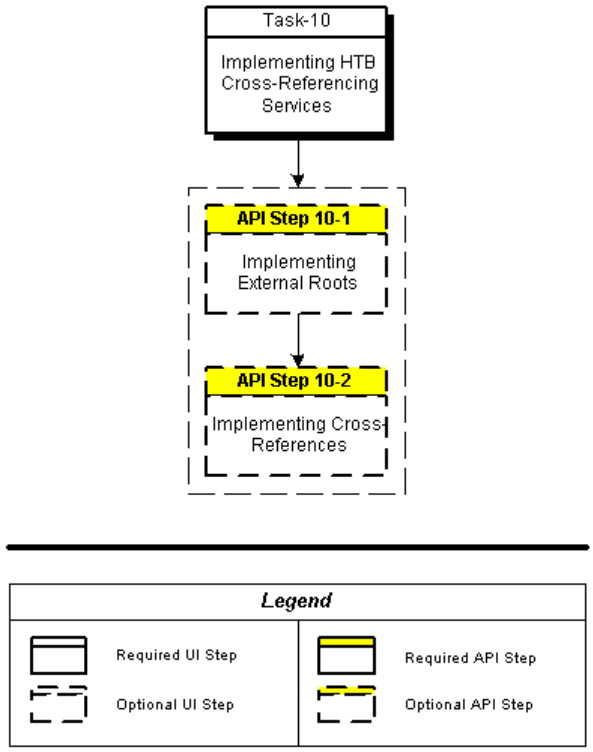
This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Procedures

Figure 4–23 provides an overview of the implementation process for HTB cross-referencing services:

Figure 4–23 Implementation Process: Cross-Referencing Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–11, HTB Implementation Procedures: Cross-Referencing

The following sections describe the implementation procedures (referenced by Figure 4–23) for Cross-Referencing services:

- Section 4.10.1, Implementing External Roots
- Section 4.10.2, Implementing Cross-References

4.10.1 Implementing External Roots

Each OID that will be configured as the root part of an external instance identifier in a cross-reference must first be registered with HTB using the methods provided by `XRefService`.

Steps

1. Use `XRefService.createRootOID` to register each of the external roots that will be used to cross-reference objects in messages. You can specify an assigning authority name with each OID to provide a human readable description. Only OIDs registered in this step can be used as the roots of external identifiers in the next step.
2. Use `XRefService.updateRootOID` to update information about the external roots.

4.10.2 Implementing Cross-References

Cross-references can be set up for the following objects:

- `ClinicalAct`
- `Encounter`
- `Material`
- `MedicalRecord`
- `StaffMember`
- `StaffPosition`
- `CareSite`
- `Organization (OrgUnit)`
- `Person`

Steps

1. Use Table 4–96 to select the appropriate method to set up cross-references based on the type of object.

   If setting up cross-references for Inbound Messaging, note that setup is necessary only for referenceable objects that cannot be created by messages. This always includes care sites and organization units, because they cannot be created by messages.

Table 4–96 Cross-Referencing: Creation Methods by Object

Object Type	Cross-Reference Creation Method
ClinicalAct	ClinicalAct.setClinicalActXRefs
Encounter	Encounter.setEncounterCrossReferences
Material	Material.setMaterialXRefs
MedicalRecord	MedicalRecord.setMedicalRecordXRefs
StaffMember	StaffMember.setXRefs
StaffPosition	StaffPosition.setXRefs
Caresite	CareSite.setCareSiteXRefs
OrgUnit	OrgUnit.setExternalOrgId
Person	CorrelatePersonService.createCorrelateData

2. Use Table 4–97 to determine which method to use to query an object and its cross-references using external identifiers.

Table 4–97 Cross-Referencing: Query Methods by Object

Object Type	Cross-Reference Query Method
`ClinicalAct`	▪ `ClinicalActService.getClinicalActByExternalId` ▪ `ClinicalActService.findClinicalActXRefByII`
`Encounter`	▪ `Encounter.getEncounterByExternalIdentifier` ▪ `EncounterService.findEncounterXRefsByInstanceIdentifier`
`Material`	▪ `ClinicalActService.getMaterialByExternalId` ▪ `ClinicalActService.findMaterialXRefByII`
`MedicalRecord`	▪ `PatientService.getMedicalRecordByExternalId` ▪ `PatientService.findMedicalRecordXRefByII`
`Staff Member`	▪ `StaffService.getStaffMemberByExternalId` ▪ `StaffService.findStaffMemberXRefByII`

Table 4–97 (Cont.) Cross-Referencing: Query Methods by Object

Object Type	Cross-Reference Query Method
StaffPosition	■ StaffService.getStaffPositionByExternalId ■ StaffService.findStaffPositionXRefByII
CareSite	■ CareSiteService.getCareSiteByExternalId ■ CareSiteService.findCareSiteXRefByII
OrgUnit	■ OrgUnitService.getOrgUnitByExternalId
Person	■ CorrelatePersonService.findPersonByExternalId

4.11 Implementing ID Type Services

Identifications store legal identification information for a person or organization within HTB, and each identification is associated with an ID Type. ID Types provide information about the nature of the identification and the assigning organization. Identifications are different from external identifiers—they do not uniquely identify either persons or organizations.

> **Note:** When HTB receives HL7 messages with person information, it expects the sending enterprise to generate a unique person identifier that is not the person's social security number, Medicaid ID, or other person identification. Typically this type of identifier is created by an Enterprise Master Person Index (EMPI) system. If an enterprise must use a person identification (such as Medicaid ID) as the external cross-reference for the person, HTB expects an enterprise-specific root to be sent with the ID configuration. This takes the place of a *global identifier* created by an EMPI system. If a Medicaid ID is used both as the external identifier for a person as well as an identification, it must be sent as *two separate instance identifiers for the person*—one with the root representing the IdType so that it can be treated as an identification, and another with the enterprise-specific root so that it can be used as an external identifier.

Within the HTB Applications Programming Interface, ID Type Services provide complete ID Type maintenance functions, including create, update, delete, and several `find` methods.

ID Type information is stored within the `IDType` value object, including the following attributes:

- Name [Name]
- Type Code [TypeCode]
- Organization ID [OrgId]
- Organization Name [OrgName]
- Identification Description [Description]
- Root OID [RootOid]

These attributes are defined in *Oracle Javadoc for HTB* (`IdType` interface); use the `IdTypeService` to maintain the value object (Table 4–51).

Reference

- *Oracle Javadoc for HTB*
- Appendix F, Seeded ID Types
- Table 4–98 lists the principal ID Type service and methods:

Table 4–98 Service and Methods: ID Type Services

Level	Detail
`Package`	`oracle.apps.ctb.configuration`
`Class`	`IdTypeService`
`Methods`	▪ `createIdType` ▪ `deleteIdType` ▪ `getIdType` ▪ `getIdTypeByName` ▪ `getIdTypeByRootOid` ▪ `getIdTypes` ▪ `updateIdType`

Prerequisites

- Implementing External Roots: `RootOids` that are used in `IdTypes` must have been registered using the `XRefService` prior to implementing `IdTypes`.
- Implementing Organizations: Organizations that are used in creating `IdTypes` must be implemented before creating the `IdTypes` themselves.

See Also:

- Section 4.3, Implementing Organizations
- Section 4.10.1, Implementing External Roots

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility you defined in Section 4.1.4.

Navigation

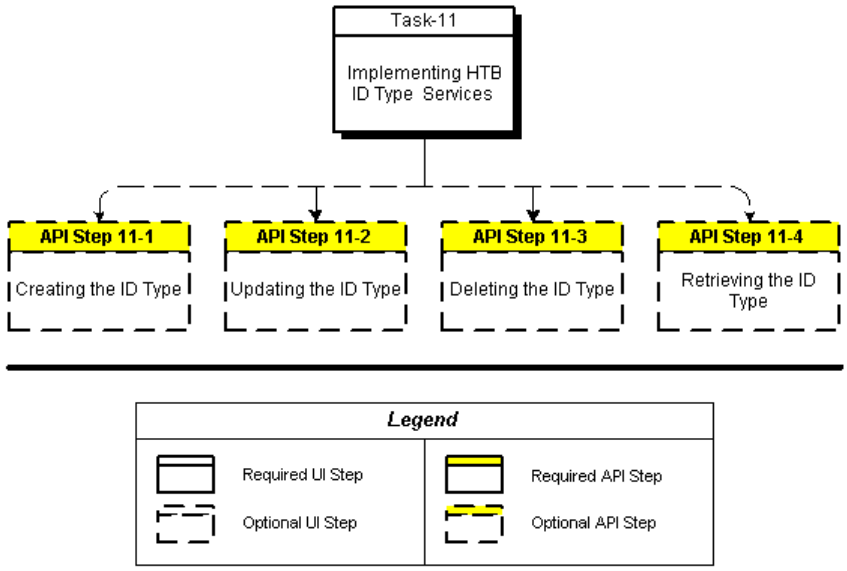
This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Procedures

Figure 4–24 provides an overview of the implementation process for ID Type Services:

Figure 4–24 Implementation Process: ID Type Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–12, HTB Implementation Procedures: ID Type Services

The following sections describe the implementation process (referenced in Figure 4–24) for ID Type Services:

- Section 4.11.1, Creating the ID Type
- Section 4.11.2, Updating ID Type
- Section 4.11.3, Deleting ID Type
- Section 4.11.4, Retrieving ID Types

4.11.1 Creating the ID Type

Use the `createIdType` method to create a new ID type. Either the organization ID or organization name must be provided. If no organization ID is provided, but the organization name is provided and an organization with such name exists, the organization ID of this organization is used to populate the organization ID field. Otherwise, a new organization is automatically created, and the newly generated organization ID is entered into this field.

4.11.2 Updating ID Type

Use the `updateIdType` method to update an existing ID Type.

4.11.3 Deleting ID Type

Use the `deleteIdType` method to delete an existing ID Type.

4.11.4 Retrieving ID Types

4.11.4.1 Getting ID Type

Use the `getIdType` method to find an existing ID Type, given its type code and ID-issuing authority's organization identifier.

4.11.4.2 Getting ID type by Name

Use the `getIdTypeByName` method to find an existing ID Type, given its name.

4.11.4.3 Getting ID type by Root OID

Use the `getIdTypeByRootOid` method to find an existing ID Type, given its root OID.

4.11.4.4 Getting ID Types

Use the `getIdTypes` method to find all existing ID Types.

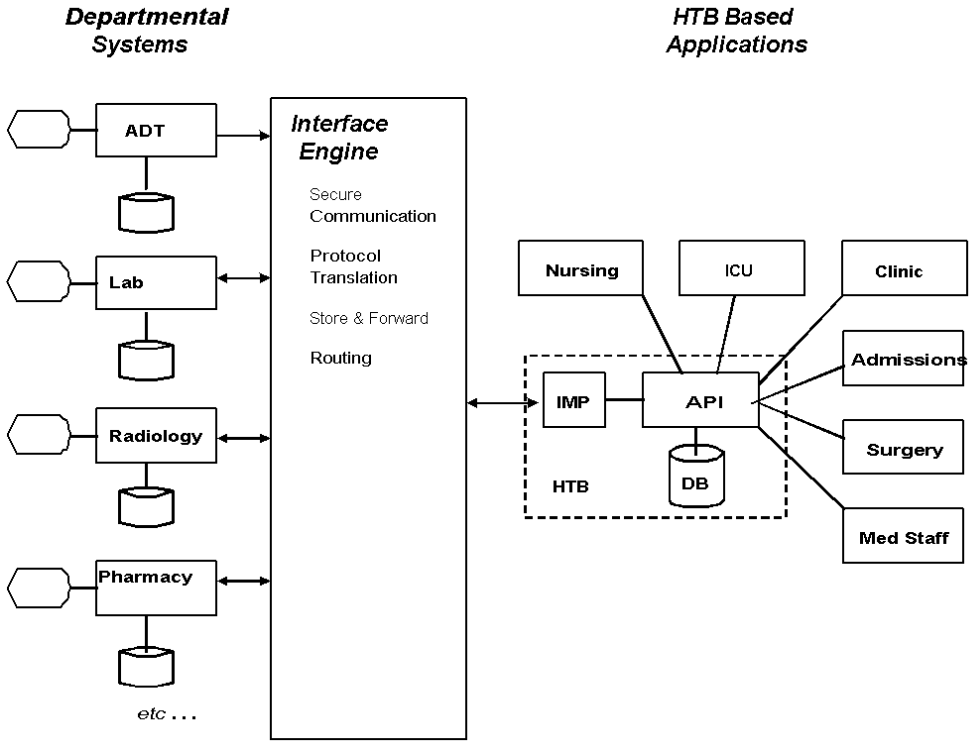
4.12 Implementing Inbound Messaging Services

Healthcare enterprises typically have a number of departmental systems such as ADT, diagnostic departments, pharmacy, and others that may be acquired from multiple vendors. Such systems require messaging to communicate events and request actions from applications throughout the enterprise.

The two principal components of system messaging are Inbound Messaging Services, described by this section, and Outbound Messaging (Section 4.13).

In Figure 4–25, the ADT system might be responsible for registering and admitting patients. After updating its own database, ADT sends an HL7 message to an interface engine that in turn routes the message to HTB and to other systems within the enterprise. HTB maintains this patient data in a clinical data repository, available to applications written against the HTB Platform.

Figure 4–25 Typical Application Topology (IMP)



The Inbound Message Processor (IMP) provides message interpretation, persistence, and acknowledgement services to HTB applications for processing inbound messages from external systems. IMP supports XML formatted inbound messages that conform to the HL7 version 3 messaging standard.

Notes About the Interface Engine:

- An external interface engine that handles HL7 Version 3 message translation and routing must be implemented for IMP to function; an interface engine is not included with HTB.
- For any messaging environment, a single interface engine is required, although multiple interface engines can be implemented.
- **Processing:** Upon receipt of a message from the sending application (the source of the message), IMP synchronously processes the message into HTB and returns (i) an Application Acknowledgment (AA) or (ii) an Application Error (AE). The interface engine is required to accept and handle all acknowledgements.
- **Application Acknowledgement (AA):** An AA response indicates that the message was successfully processed and persisted in HTB.
- **Application Error (AE):** An AE response indicates an error reported by HTB, including error information in functional message segments (error type code, error detail code, free text). It is the responsibility of the interface engine to determine if the acknowledgement message is returned to the sending system or if the message should be resent to HTB or skipped (abandoning the message). IMP does not support sequence number protocol. The interface engine is responsible for assuring that messages are delivered in order.
- For more information about integrating IMP services with an interface engine, refer to documentation from your interface engine vendor.

See Also: *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i,* for information about message types supported by IMP

Reference

- *Oracle JavaDoc for HTB*
- *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i*

Prerequisites

- **Implementing Person Services:** For successful processing of an inbound message, each reference to a person in the message must either be resolved to an existing person in HTB, or a new person with that cross-reference must be created. In the case of a person object that is the focus of a person registry message, a person is always created if the data does not exist in HTB. But in the case of a person in a non-person registry message, the person is created only if such a side effect is configured (Section 4.12.4). If a side effect is not configured, referenced persons must be created (Section 4.2) before implementing IMP.
- Implementing Person Domain Service: If the use of Flexible Person Domains is configured (Section 4.2.4), the processing of Person Registry messages results in the creation of a person in the Patient domain.
- **Implementing Organizations:** Organization units of the following organization types—that are referenced by inbound messages—must exist:
  - Enterprise
  - Facility
  - Practice Setting

  Note that organizations cannot be created by messages. Accordingly, all Organization must be created using the HTB application program interface prior to implementing IMP.
- Implementing Administrative Business Services:
  - **Implementing Care Sites:** Care Sites that are referenced by inbound messages must exist. Note that care sites cannot be created by messages. Accordingly, all care sites must be created using the HTB application program interface prior to implementing IMP.

  - **Implementing Patient Registration:** For successful processing of an inbound message, each reference to a patient in the message must either be resolved to an existing medical record in HTB, or a new medical record with that cross-reference must be created. If the side effect configuration permits creation of a medical record, patient data from the message is used to create or update a medical record in HTB. If a side effect is not configured, referenced medical records must be created (Section 4.8.9) before implementing IMP. Note that patient objects from messages correspond to medical records in HTB.
  - **Implementing Staff Management:** IMP can create a staff member as the result of a message only if the staff member is the focal object of a Staff Registry message. References to all other staff members in messages must be resolved to existing staff members in HTB. These staff members must therefore be created (Section 4.8.4) before implementing IMP. Note that staff members cannot be created as the side effect of any message.

- **Implementing Master Catalogs:** Act Definitions that are referenced by clinical acts in inbound messages must exist. Note that act definitions cannot be created by messages. Accordingly, all Act Definitions must be created using the HTB application program interface prior to implementing IMP.
- Implementing Cross-Referencing: When objects are created by IMP as the result of receiving a message, HTB assigns them internal identifiers and automatically creates cross-references that relate this internal identifier to the external identifier from the message. However, for objects referenced by messages that cannot be created using messages, these cross-references must be implemented manually (Section 4.10).
- Implementing ID Type Services: Some of the instance identifiers received for persons in messages correspond to HTB Person identifications—legal identifications such as a social security number, not unique external cross-references to a person. The `IdTypes` corresponding to these identifications must be set up so that IMP does not mistake them for valid external identifiers usable for cross-referencing.
- Implementing Enterprise Terminology Services: Inbound messages can contain coded data types that include a `codeSystemName`, and code but no `codeSystemVersion`. In the absence of a version, IMP searches for the code in the default version of that terminology as configured in ETS. If no default version is found the message is rejected. Configuring default versions is critical because some terminologies may reuse codes across versions. To use the

concept with the intended meaning, ETS must explicitly know which version to use for a certain terminology.

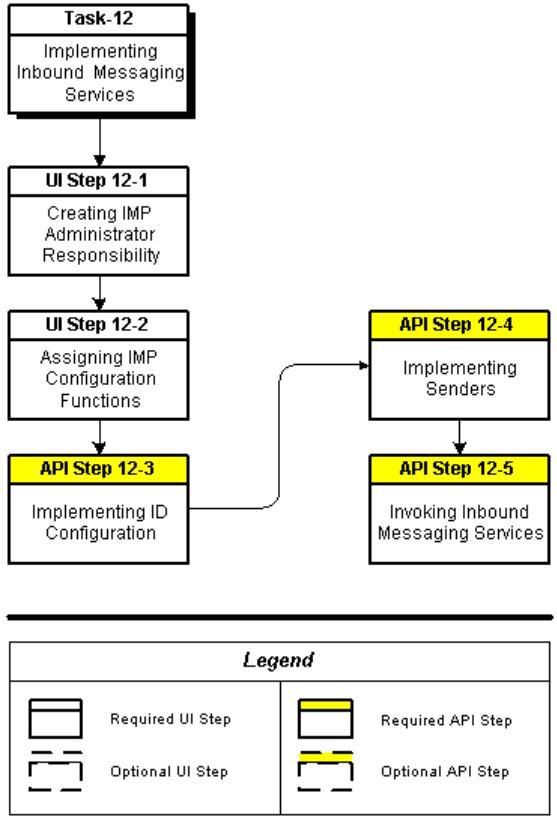
See Also:

- Section 4.2, **Implementing Person Services**
- Section 4.3, **Implementing Organizations**
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.8.4, **Implementing Staff Management**
- Section 4.8.5, **Implementing Care Sites**
- Section 4.8.9, **Implementing Patient Registration**
- Section 4.9.1, **Implementing Master Catalogs**
- Section 4.10, Implementing Cross-Referencing
- Section 4.11, Implementing ID Type Services

Procedures

Figure 4–26 provides an overview of the implementation process for Inbound Messaging Services:

Figure 4–26 Implementation Process: Inbound Messaging Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–13, HTB Implementation Procedures: Inbound Messaging Services

The following sections describe the implementation procedures for Inbound Messaging Services (referenced by Section 4–26):

- Section 4.12.1, Creating an IMP Administrator Responsibility and Assigning it to an Account
- Section 4.12.2, Assigning IMP Configuration Functions to the IMP Administrator Responsibility
- Section 4.12.3, Implementing ID Configuration
- Section 4.12.4, Implementing Senders, Receivers, Trigger Events and Side Effects
- Section 4.12.5, Invoking Inbound Messaging Services

4.12.1 Creating an IMP Administrator Responsibility and Assigning it to an Account

An *IMP Administrator Responsibility* should be created and assigned to the account that will be used to implement IMP. To perform this task, refer to the following sections:

- Section 4.1.4, Creating Responsibilities
- Section 4.1.5, Assigning Accounts to Responsibilities

4.12.2 Assigning IMP Configuration Functions to the IMP Administrator Responsibility

Table 4–99 lists functions that must be granted to the IMP Administrator Responsibility:

Table 4–99 IMP Configuration Functions

Function Name	Description
CTB_MS_CREATE_ID_CFG	Create ID configuration
CTB_MS_CREATE_SND_CFG	Create sender configuration
CTB_MS_GET_ID_CFG	Get ID configuration
CTB_MS_GET_SND_CFG	Get Sender configuration
CTB_MS_REMOVE_ID_CFG	Remove ID configuration
CTB_MS_REMOVE_SND_CFG	Remove sender configuration
CTB_MS_UPDATE_ID_CFG	Update ID configuration
CTB_MS_UPDATE_SND_CFG	Update sender configuration

See Also: Section 4.1.8, Assigning Functions to Responsibilities Conditioned by Criteria, for information about assigning IMP configuration functions to the IMP Administrator responsibility.

4.12.3 Implementing ID Configuration

Objects in HTB are uniquely identifiable, based on internal identifiers that are assigned to them at the time of their creation. When external systems send messages, they typically use a different set of identifiers to refer to the same objects. Each such identifier space is identified by a root—an ISO Object Identifier or OID. HTB provides cross-reference services to translate an external ID to an internal ID. IMP needs to know the set of external roots for each object type that is recognized by the enterprise in order to accept or reject an external reference. IMP can also be configured with an internal root for each such object type. When a sender uses the configured internal root, IMP interprets this to mean that the reference being sent is an internal (HTB) identifier and can be used without any cross-referencing. IMP ID configuration lets the administrator configure these internal and external roots for each identified object that can be included in a message.

Reference

Oracle Javadoc for HTB

Table 4–100 lists the principal ID service and methods (for IMP implementation):

Table 4–100 Service and Methods: IDs (IMP)

Level	Detail
Package	oracle.apps.ctb.message.configuration
Class	IMPConfigurationAdministrationService
Methods	■ createIdConfiguration ■ createSenderConfiguration ■ removeIdConfiguration ■ removeSenderConfiguration ■ updateIdConfiguration ■ updateSenderConfiguration
Class	IDConfigurationItem

Table 4–100 (Cont.) Service and Methods: IDs (IMP)

Level	Detail
Methods	▪ setIdentifiedObjectType ▪ setIsInternalRoot ▪ setReceiverIdentifier ▪ setRoot

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility created in Section 4.12.1, with the functions described in Section 4.12.2.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

1. **Create ID Configuration:** Use the `createIdConfiguration` method to populate the `IdConfigurationItem` object with the following information:
   - `IdentifiedObjectType`
   - `Root`
   - `IsInternalRoot`
   - `ReceiverIdentifier`

   See Also: *Oracle Javadoc for HTB*, for further information about these attributes.

2. **Update ID Configuration:** Use the `updateIdConfiguration` method to update the `IdConfigurationItem` value object.

   See Also: *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i*

4.12.4 Implementing Senders, Receivers, Trigger Events and Side Effects

When IMP receives a message, it extracts `SenderIdentifier` and `ReceiverIdentifier` from the message wrapper. It also extracts the **Trigger Event** (`TriggerEventCode`) from the control act wrapper. IMP uses its configuration to determine if the `SenderIdentifier` and `ReceiverIdentifier` are a valid combination. If they are not, the message is rejected. This configuration thus controls who is a valid **sender** and which HTB enterprises they are authorized to send messages to. This is called the `SenderConfiguration`.

Upon validation of the `SenderConfiguration`, IMP uses its configuration to determine if the `TriggerEventCode` is valid for the `SenderConfiguration`. If it is not configured for that `SenderConfiguration`, IMP rejects the message. This configuration thus controls what types of trigger events a sender is permitted to send to a **receiver**. This is called the `SenderTriggerConfiguration`.

Upon validation of the `SenderConfiguration` and `SenderTriggerConfiguration` combination, IMP processes the message payload. The focal object is created or updated in the HTB Repository. However, for non-focal objects, IMP inspects its **side effect** configuration to determine its behavior. You can configure IMP to let each non-focal object type create or not create the object if *it is not* present in the repository, and to update or not update the object *if it is* present in the repository. This configuration of side effects is called the `SenderReferenceConfiguration`. Note that if a non-focal object is not present in the repository and the `SenderReferenceconfiguration` prohibits its creation, the message is rejected.

Reference

Oracle Javadoc for HTB

Table 4–101 lists the principal service and methods for implementing senders:

Table 4–101 Service and Methods: Senders, Receivers, Trigger Events and Side Effects

Level	Detail
`Package`	`oracle.apps.ctb.message.configuration`
`Class`	`IMPConfigurationAdministrationService`

Table 4–101 (Cont.) Service and Methods: Senders, Receivers, Trigger Events and Side Effects

Level	Detail
Methods	▪ createSenderConfiguration ▪ removeSenderConfiguration ▪ updateSenderConfiguration
Class	SenderConfiguration
Methods	▪ setReceiverIdentifier ▪ setSenderIdentifier ▪ setSenderTriggerConfiguration
Class	SenderTriggerConfiguration
Methods	▪ setTriggerEventCode ▪ setSenderReferenceConfiguration

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility created in Section 4.12.1, with the functions described in Section 4.12.2.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

1. **Create Sender Configuration:** Use the `createSenderConfiguraton` method to configure valid combinations of **sender**s and **receiver**s. Senders are identified by the `SenderIdentifier` and receivers by the `ReceiverIdentifier`. Note that the `ReceiverIdentifier` must be a valid internal identifier for an HTB enterprise; only one receiver can be configured.

   For each configured sender and receiver combination, an array of `SenderTriggerConfiguration` objects can be configured. Each

SenderTriggerConfiguration object identifies one trigger event that the sender can transmit to the receiver and the **side effect**s that are permitted on a message of that trigger event.

2. **Create Sender Trigger Configuration:** Use the setSenderTriggerConfiguration method to populate a SenderTriggerConfiguration object and assign it to one or more SenderConfigurations.

   Each SenderTriggerConfiguration must be populated with a valid TriggerEventCode. This code must belong to the list of valid membership codes contained in the CTB_MS_TRIGGER_EVENT concept list.

   Note: If an HTB-based application manages care site housekeeping and maintenance through direct calls to the CareSiteMgmtService interface, IMP should not be configured to support the bed status trigger event (PRPA_TE000220). Conflicts between the activities managed through the interface and those managed through inbound messages may result in a *duplicate activity error* within IMP. This can occur when the HTB-based application opens multiple concurrent housekeeping or maintenance activities for a single care site. IMP con only manage a single activity at a time through the current HL7 v3 messages.

   For each SenderTriggerConfiguration, an array of SenderReferenceConfiguration objects can be configured. These objects determine what side effects are permitted when a message with this trigger event is received. Note that because the SenderTriggerConfiguration is ultimately associated with a Senderconfiguration, these side effects are specific to a particular combination of sender, receiver and trigger event.

3. **Create Sender Reference Configuration:** Use the setSenderReferenceConfiguration method to populate a SenderReferenceConfiguration object and assign it to a SenderTriggerConfiguration.

   Each SenderReferenceConfiguration must be populated with a valid ReferencedObjecType. This code must belong to the list of valid membership codes contained in the CTB_MS_REF_OBJECT_TYPE concept list.

For each `ReferencedObjectType`, a `CreatePermission` and an `UpdatePermission` can be configured. `CreatePermission` controls if a non-focal object of that type in a message is created as a side effect if it does not already exist in the repository. `UpdatePermission` controls if a non-focal object of that type is updated with message information if it already exists in the repository.

If the `ReferencedObjectType` is *Person* and the `UpdatePermission` is set to *Y* (Yes), you can also configure how IMP handles information about the contacts, **guarantor**s, and **primary care physician**s associated with a non-focal person object that is being updated. This is achieved by setting an array of `SenderReplaceConfiguration` objects on the `SenderReferenceConfiguration`.

Table 4–102 lists the object types that can have create and update side effect configuration in messages that belong to a particular domain. Note that the information in this table is applicable only if inbound messages are constructed in accordance with the XSDs published in the *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification.* The set of trigger events that correspond to each message domain is defined in the conformance specification.

Table 4–102 Object Types by Domain: Side Effects

Message Domain	Object Type[1]
Person Registry	PERSON
Person Merge	PERSON
Staff Registry	PERSON
Encounter Event	▪ ENCOUNTER_EVENT ▪ OBSERVATION_EVENT ▪ PATIENT ▪ PERSON
Bed Status	NA
Observation Event Intolerance	▪ ENCOUNTER_EVENT ▪ OBSERVATION_EVENT ▪ PATIENT ▪ PERSON
Encounter Appointment	▪ ENCOUNTER_EVENT ▪ OBSERVATION_EVENT ▪ PATIENT ▪ PERSON

Table 4–102 (Cont.) Object Types by Domain: Side Effects

Message Domain	Object Type[1]
Diet Request	■ ENCOUNTER_EVENT ■ OBSERVATION_EVENT ■ PATIENT ■ PERSON ■ CONDITIONPROBLEM ■ MATERIAL ■ SUBSTANCEADMIN_ORDER ■ DIET_ORDER
Condition Problem	■ ENCOUNTER_EVENT ■ OBSERVATION_EVENT ■ PATIENT ■ PERSON ■ CONDITIONPROBLEM
Medication Supply Event	■ ENCOUNTER_EVENT ■ OBSERVATION_EVENT ■ PATIENT ■ PERSON ■ SUBSTANCEADMIN_ORDER ■ MATERIAL
Substance Administration Event	■ ENCOUNTER_EVENT ■ OBSERVATION_EVENT ■ PATIENT ■ PERSON ■ MATERIAL ■ SUBSTANCEADMIN_ORDER
Substance Administration Order	■ ENCOUNTER_EVENT ■ OBSERVATION_EVENT ■ OBSERVATION_EVENT_CRITERION ■ PATIENT ■ PERSON ■ CONDITIONPROBLEM ■ MATERIAL ■ SUPPLY_ORDER ■ SUBSTANCEADMIN_ORDER

Table 4–102 (Cont.) Object Types by Domain: Side Effects

Message Domain	Object Type[1]
Supply Request	▪ ENCOUNTER_EVENT ▪ OBSERVATION_EVENT ▪ PATIENT ▪ PERSON ▪ CONDITIONPROBLEM ▪ MATERIAL
Observation Event	▪ ENCOUNTER_EVENT ▪ OBSERVATION_EVENT ▪ PATIENT ▪ PERSON ▪ CONDITIONPROBLEM ▪ PROCEDURE_ORDER ▪ OBSERVATION_ORDER
Observation Order	▪ ENCOUNTER_EVENT ▪ OBSERVATION_EVENT ▪ OBSERVATION_EVENT_CRITERION ▪ PATIENT ▪ PERSON ▪ CONDITIONPROBLEM ▪ OBSERVATION_ORDER
Procedure Order	▪ ENCOUNTER_EVENT ▪ OBSERVATION_EVENT ▪ OBSERVATION_EVENT_CRITERION ▪ PATIENT ▪ PERSON ▪ CONDITIONPROBLEM ▪ PROCEDURE_ORDER

[1] Object types that can have *create* and *update* side effect configuration in messages directed to the associated domain.

See Also: *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i*

4. **Create Sender Replace Configuration:** Use the `setSenderReplaceConfiguration` method to populate a `SenderReplaceConfiguration` object and assign it to a `SenderReferenceConfiguration`.

   Each `SenderReplaceConfiguration` must be populated with a valid `ReferencedObjectType`. This code must belong to the list of valid membership codes contained in the `CTB_MS_RPL_OBJECT_TYPE` concept list.

   For each `ReferencedObjectType`, a `ReplacePermission` can be configured. When the parent `SenderReferenceConfiguration` grants `UpdatePermission` to a *Person* object type, the `SenderReplaceConfiguration` controls whether the objects of the specified type are appended to the existing list of person-related objects of that type, or they entirely replace the list in the repository.

   Exceptions:

   The following exceptions apply to sender reference configuration:

   - Sender Reference Configuration is only applicable to objects that are distal to the focal class in the XML tree; it is not applicable to the message wrapper, control act wrapper, objects that occur proximal to the focal class in the message payload, or the focal class itself.
   - In patient administration messages, if the focal encounter has a `CoverageMother` act relationship, sender reference configuration is not applicable to the message tree at the target end of this act relationship. This means that the mother's encounter, the mother's patient responsibility, and the mother person should all exist prior to receiving the message.
   - In patient administration messages, the sender reference configuration for the patient object is only used for the participant of the `SubjectPatient` participation of the focal encounter. When `SubjectPatient` participations occur in other parts of the message, they identify the same patient as the `SubjectPatient` participation of the focal encounter.
   - In patient administration messages, the sender reference configuration for the patient object is only used for the participant of the *Subject* participation of the focal encounter. When *Subject* participations occur at other parts of the message, they identify the same patient as the *Subject* participation of the focal encounter.
   - In observation event messages, sender reference configuration is not applicable to the message tree at the target end of `Appendage and ReplacementOf act` relationships. This means that the original

observations being appended to and the observation being replaced must exist in HTB prior to receiving the appendage or replacement.

- In clinical messages, sender reference configuration for Encounter object is limited. When the encounter is created using data in the clinical message (side effect processing), the encounter is only associated with the focal clinical act. Any references to the encounter from within acts that are related to the focal act are not persisted. If the encounter exists prior to the processing of the clinical message, all relationships to the preexisting encounter object are persisted.

  See Also: *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i*

5. **Update Sender Configuration:** Use the following methods to update or remove sender configurations:
   - `updateSenderConfiguration`
   - `removeSenderConfiguration`

4.12.5 Invoking Inbound Messaging Services

Reference

- *Oracle Javadoc for HTB*
- *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i*

Table 4–103 lists the principal IMP service and methods:

Table 4–103 Service and Methods: IMP

Level	Detail
`Package`	`oracle.apps.ctb.message.improcessor`
`Class`	`IMPService`
`Methods`	▪ `processMessage`
`Class`	▪ `Result`
`Methods`	▪ `getResponseXML` ▪ `getStatus`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the seeded Inbound Message Processor Responsibility.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Steps

1. Use the Service Locator to access the IMP Service.
2. Use the `processMessage` method with an HTB compliant instance[1] as a parameter to invoke message processing services; a Result object is returned.
3. Use the following methods to inspect the result of processing the message:
   - `getResponseXML`
   - `getStatus`

   See Also:

   - *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i*, for information about message types supported by IMP
   - Example A–1 (Appendix A), Sample Java Code to Create a Session

[1] IMP supports XML formatted inbound messages that conform to the HL7 version 3 messaging standard; see the *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specifications, Release 11i*, available from Oracle *MetaLink*, for further information.

4.13 Implementing Outbound Messaging Services

Background

Healthcare enterprises typically have a number of departmental systems such as ADT, diagnostic departments, pharmacy, and others that may be acquired from multiple vendors. Such systems require messaging to communicate events and request actions from applications throughout the enterprise.

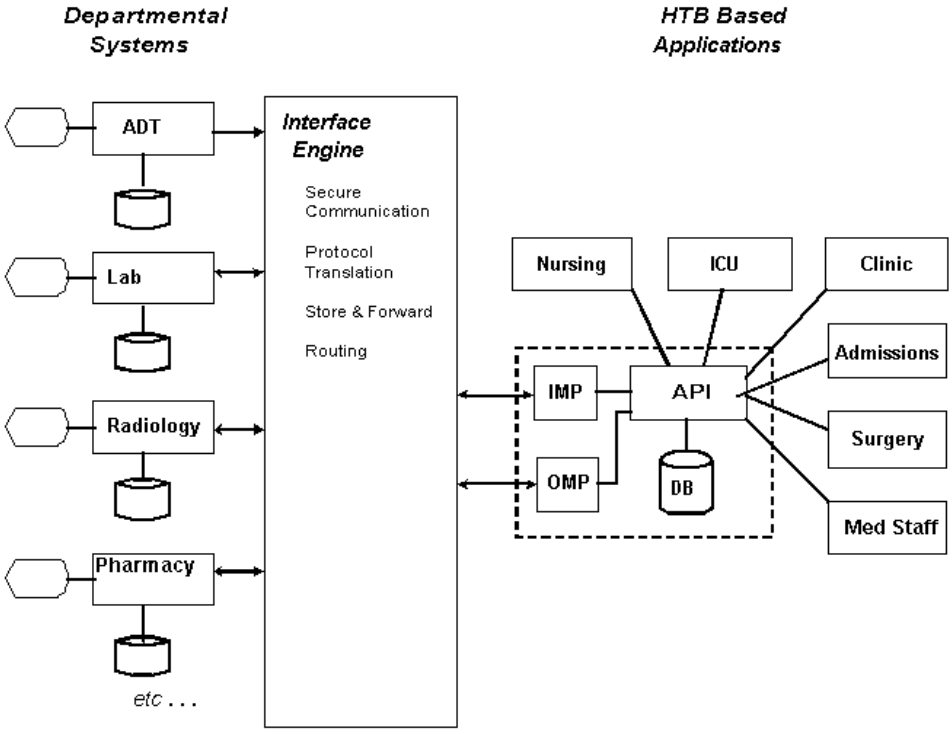
The two principal components of system messaging are Inbound Messaging Services (Section 4.12), and Outbound Messaging Services, described by this section.

Overview: HTB Outbound Messaging

The Outbound Message Processor (OMP) lets you generate and send outbound messages from the HTB Platform. Data extracted from the HTB Data Repository is formulated and mapped to HL7 v3 conformant messages, which are queued and transmitted to an external interface engine that routes messages to appropriate external systems, subject to local configuration rules.

In Figure 4–27, the ADT system might be responsible for registering and admitting patients. After updating its own database (HTB), the ADT system sends an HL7 message to the external interface engine that in turn routes the message to other systems in the enterprise. HTB maintains this patient data in a clinical data repository, available to applications written against the HTB Platform.

Figure 4–27 Typical Application Topology (OMP)



OMP provides message formulation, transmission, acknowledgement, error processing, and message logging—all accessible to user-developed applications through an API call interface. OMP supports XML formatted outbound messages that conform to the HL7 version 3 messaging standard.

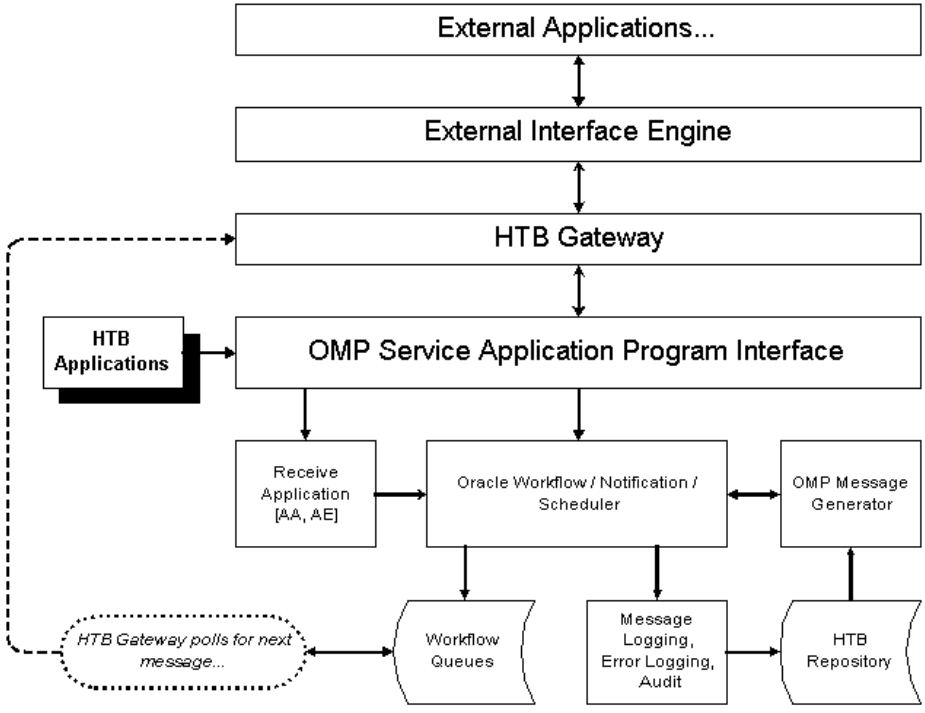
For details about integrating OMP services with an external interface engine, refer to documentation from your interface engine vendor.

> **See Also:** *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i*

Architecture

Figure 4–28 depicts the architecture of the Outbound Message Processor:

Figure 4–28 OMP Architecture



HTB applications initiate outbound messages by making service calls to the OMP Application Program Interface. Such service calls raise a business event with Oracle Workflow that executes a preconfigured workflow subscribed to the event.

The workflow process, which may include a scheduled delay, triggers the OMP Message Generator to formulate a message. It determines which message to generate, populates it with information extracted from the HTB Repository, and formulates an HL7 version 3 conformant message. Workflow places the outbound

message in a queue maintained by Oracle Advanced Queuing (imbedded with Oracle Workflow), and logs all generated messages to the HTB Repository.

The HTB Gateway polls the workflow queue to determine the next message for each receiving organization and initiate message transmission to the interface engine.

Notes:

- An external interface engine that handles HL7 version 3 format translation and message routing must be implemented in the messaging environment for OMP to function; an interface engine is not included with HTB.
- For any messaging environment, a single interface engine is required. However, multiple interface engines can be implemented.

Upon receipt of a message acknowledgement from the receiving application (the target of the message), the interface engine calls the OMP service interface to log it. There are three possible responses from a receiving application: (i) Application Acknowledgement (AA), (ii) Application Error (AE), or (iii) Application Reject (AR). However, OMP only accepts AA or AE acknowledgement codes. The interface engine is required to resolve all AR acknowledgements to either an AA or an AE acknowledgement.

An AA response indicates that the message was successfully transmitted and received by the target application. OMP logs the response and attaches it to the original message, closing it out.

An AE response indicates an error reported by the target application, including error information in functional segments (includes an error type code, error detail code and a free text message). OMP logs the error response, attaches it to the original message, and initiates a workflow notification, which calls error processing. OMP continues to suspense the message until the error condition is resolved, which may require human intervention. Resolution means correcting and resending the message or skipping it (abandoning the message).

An AR response indicates that the outbound message has been rejected, for reasons unrelated to its content or format (system or network down, network transmission errors...). For most such problems, the receiving system may be able to accept the

message at a later time. The sending system or interface engine must decide on an application-specific basis whether the message should be sent again. Ultimately, the AR is resolved to either an AA (upon successful retransmission) or an AE—which thence generates a call to error processing.

Reference

Oracle JavaDoc for HTB

Prerequisites

- **Implementing Organizations:** Organization units of the following organization types—that are referenced by inbound messages—must exist:
  - Enterprise
  - Facility
  - Practice Setting

  Cross-references for these organizations must also exist. Note that organizations cannot be created by messages. Accordingly, all Organization must be created using the HTB application program interface prior to implementing OMP.
- **Implementing Security Services**
- **Implementing Person Services:** Persons that are referenced by outbound messages must exist, and cross-references for such persons must also exist. Note that persons may also be created via messaging using Person Registry messages or as side effects of non-Person Registry messages (Section 4.12.4).
- **Implementing Enterprise Terminology Services**
- Implementing Administrative Business Services:
  - **Implementing Care Sites:** Care Sites that are referenced by outbound messages must exist, and cross-references for such care sites must also exist.
  - **Implementing Patient Registration:** Patients and medical records that are referenced by outbound messages must exist, and cross-references for such medical records must also exist.
  - **Implementing Staff Management:** Staff members that are referenced by outbound messages must exist, and cross-references for such staff members must also exist. Note that staff members may also be created via messaging using Staff Registry messages.

  - **Implementing Patient Scheduling**
  - **Implementing Encounter Management**
- **Implementing Clinical Business Services:**
  - **Implementing Master Catalogs:** Act Definitions that are referenced by clinical acts in outbound messages must exist.
  - Implementing Medication Orders
- Implementing *Oracle Workflow*
- Implementing *Oracle Advanced Queuing*
- Implementing an External Interface Engine

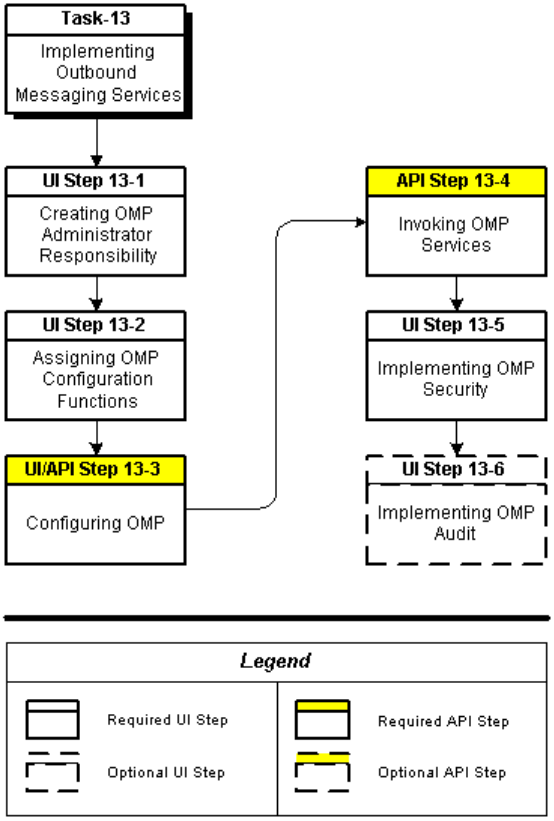
See Also:

- Section 4.1, Implementing Security Services
- Section 4.2, **Implementing Person Services**
- Section 4.3, **Implementing Organizations**
- Section 4.5, Implementing Enterprise Terminology Services
- Section 4.7, Implementing Consent Services
- Section 4.8.4, **Implementing Staff Management**
- Section 4.8.5, **Implementing Care Sites**
- Section 4.8.9, **Implementing Patient Registration**
- Section 4.8.11, Implementing Patient Scheduling
- Section 4.8.12, Implementing Encounter Management
- Section 4.9, Implementing Clinical Business Services
- Section 4.9.1, **Implementing Master Catalogs**

Procedures

Figure 4–29 provides an overview of the implementation process for Outbound Messaging Services:

Figure 4–29 Implementation Process: Outbound Messaging Services



See Also:

- Figure 3–1, Implementation Task Process: HTB Platform
- Table 3–1, HTB Implementation Tasks Summarized
- Table 3–14, HTB Implementation Procedures: Outbound Messaging Services

The following sections describe the implementation procedures for Outbound Messaging Services (referenced by Figure 4–17):

- Section 4.13.1, Creating and Assigning an OMP Administrator Responsibility
- Section 4.13.2, Assigning OMP Configuration Functions
- Section 4.13.3, Configuring OMP
- Section 4.13.4, Invoking Outbound Messaging Services
- Section 4.13.5, OMP Security
- Section 4.13.6, **OMP Audit**

4.13.1 Creating and Assigning an OMP Administrator Responsibility

You must create an OMP Administrator responsibility and assign it to the account that will be used to implement OMP. To perform this task, refer to the following sections:

- Section 4.1.4, Creating Responsibilities
- Section 4.1.5, Assigning Accounts to Responsibilities

4.13.2 Assigning OMP Configuration Functions

You must assign the OMP configuration functions listed in Table 4–104 to the OMP Administration responsibility:

Table 4–104 OMP Configuration Functions

Function Name	Description
CTB_MS_CREATE_RCV_CFG	Create Receiver Configuration
CTB_MS_UPDATE_RCV_CFG	Update Receiver Configuration
CTB_MS_REMOVE_RCV_CFG	Remove Receiver Configuration
CTB_MS_CREATE_RCV_TRG_CFG	Create Trigger Configuration
CTB_UPDATE_RCV_TRGCFG	Update Trigger Configuration
CTB_MS_REMOVE_RCV_TRGCFG	Remove Trigger Configuration
CTB_MS_CREATE_RCV_VOCAB_CFG	Create Vocabulary Configuration
CTB_MS_UPDATE_RCV_VOCAB_CFG	Update Vocabulary Configuration
CTB_MS_REMOVE_RCV_VOCAB_CFG	Remove Vocabulary Configuration

See Also: Section 4.1.8, Assigning Functions to Responsibilities Conditioned by Criteria, for information about assigning OMP configuration functions to the OMP Administrator responsibility.

4.13.3 Configuring OMP

When OMP receives a request to generate and send an outbound message, it makes several processing decisions that are configuration dependent. OMP configuration requirements are described in the following subsections:

- Section 4.13.3.1, Creating Users and Assigning Responsibilities
- Section 4.13.3.2, Creating HTB Organization
- Section 4.13.3.3, Configuring OMP Profile Options
- Section 4.13.3.4, Configuring Receiving Organizations and Trigger Events
- Section 4.13.3.5, Configuring Receiver Vocabulary
- Section 4.13.3.6, Configuring Clinical Concepts
- Section 4.13.3.7, Configuring Oracle Advanced Queuing
- Section 4.13.3.8, Configuring Oracle Workflow
- Section 4.13.3.9, Configuring Business Event System
- Section 4.13.3.10, Setting up HTB Gateway

4.13.3.1 Creating Users and Assigning Responsibilities

OMP must access the HTB Repository to retrieve information to construct outbound messages.

Steps

1. Create an OMP Retrieval User Account, and assign both the seeded Outbound Message Processor responsibility and the Healthcare Security Administrator responsibility to this account. You can also grant the Workflow Administrator Web Applications responsibility to access the Workflow Status Monitor.
2. Create another OMP Administrator user account; assign the OMP Administrator responsibility to this user account—this is the new OMP Administrator.

See Also:

- Section 4.1.3, Creating Accounts
- Section 4.1.5, Assigning Accounts to Responsibilities
- Section 4.1.8, Assigning Functions to Responsibilities Conditioned by Criteria
- Section 4.1.10, Implementing Organization Based Security

4.13.3.2 Creating HTB Organization

OMP uses the HTB organization to populate the OID (Object Identifier) root on all internal identifiers when sent in outbound messages. If IMP and OMP are both configured in the same HTB instance, the IMP sender and organization configuration and the OMP HTB organization configuration must both use the same OID for HTB.

Steps

Create a new organization with the `shortName` *HTB* (Section 4.3).

4.13.3.3 Configuring OMP Profile Options

Set the following profile options to define the HTB Organization OID and HTB Organization ID:

- `CTB_OMP_HTB_OID`: This profile option defines the HTB Organization OID that is captured as part of the outbound messages to identify the originating organization. The profile option value should be set to the `ExternalOrgUnitID` attribute (external organization identifier) of the HTB organization created using the short name of HTB.
- `CTB_OMP_HTB_ORG_ID`: This profile option defines the HTB Organization ID (Internal HTB ID of the organization) needed while generating the outbound message. The profile option value should be set to the `OrgUnitID` attribute (internal HTB organization identifier) of the HTB organization created using the short name of HTB.

See Also:

- Section 4.3.1.1, Creating Organization Units
- Section 4.4, Implementing Profile Option Services

4.13.3.4 Configuring Receiving Organizations and Trigger Events

Each outbound message generated by OMP includes a receiving organization and a receiving system. For a particular organization, you can configure the following:

- Receivers
- Triggers
- Vocabulary
- Clinical Concepts

For any combination of configurations, there must be a unique receiving organization. If multiple applications are running on a particular system and all of the OMP configurable parameters used by the applications are the same, only one receiving organization is required. Conversely, if multiple applications running on a single system require different OMP configurations, different receiving organizations must be set up.

Notes:

The following cannot be changed after initial setup:

- `ReceiverConfiguration.ackType`
- `ReceiverConfiguration.waitFlag`
- `ReceiverConfiguration.queueLocationName`

If any of these values must be changed after initial setup:

- Flush all outbound messages through the advanced queues and workflow to the interface engine.
- Set up a new receiving organization to process future messages; *do not use the old receiving organization.*

Steps

Creating New Receiving Organizations

To define required organizations, including position in organization hierarchy (analysis step) for implementation:

1. Create new receiving organizations (Section 4.3). Please note all the receiving organizations created as part of this step should be *Internal Organizations.*

2. Set up Purpose Codes (Section 4.1). Purpose codes document the reason that **protected health information** is being communicated electronically to an external system (Section 4.7).
3. Assign Purpose Codes to receiving organizations (called *Receivers*)—one purpose code to each Receiver.
4. Set up Receiver configuration. Once the receiving organizations are created as Internal Organizations, purpose codes are set up. Use the following methods to create receiving organizations:
   - `createReceiverConfiguration`
   - `removeReceiverConfiguration`
   - `updateReceiverConfiguration`

   Notes:
   - There must be a unique queue name per interface—use the queue names defined in Section 4.13.3.7.
   - Define Application Role—Acknowledgement Type, Wait Flag.
   - Define Purpose Code using the `ReceiverConfiguration.setPurposeCode` method.

Creating New Receiver Trigger Configuration

For each trigger event sent to each receiving organization there are several parameters that control OMP processing:

- Message Delivery Scheduling:

  The OMP configuration supports configuring a particular receiving organization to receive messages (i) immediately or (ii) based on the start date and time of the message.

Note: Do not set a delay for trigger events that activate an act. The only trigger events that should use this configuration are those that support the following state transitions:

- Null to active
- New to active

If this requirement is not met, OMP could send out a cancel message to a receiver before sending the activate message.

- Interaction IDs

  Based on the information entered for `acknowledgementtype` and `WaitFlag`, create a unique `interactionID` to represent the communications between HTB and the receiving organization that is performing an HL7 application role.

  See Also: *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specification, Release 11i—Patch Set D*, for a list of trigger events supported by HTB; available at *OracleMetaLink*.

Reference

Oracle Javadoc for HTB

Table 4–105 lists the principal OMP Receiver Configuration Service and methods:

Table 4–105 Service and Methods: OMP Receiver Configuration

Level	Detail
Package	`Oracle.apps.ctb.message.configuration`
Class	`OMPConfigurationAdministrationService`
Methods	▪ `createReceiverConfiguration` ▪ `createTriggerConfiguration` ▪ `removeReceiverConfiguration` ▪ `removeTriggerConfiguration` ▪ `updateReceiverConfiguration` ▪ `updateTriggerConfiguration`

Table 4–105 (Cont.) Service and Methods: OMP Receiver Configuration

Level	Detail
Class	`OMPConfigurationService`
Methods	▪ `getReceiverConfiguration` ▪ `getTriggerConfiguration` ▪ `getTriggerInfo`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility created in Section 4.13.3.1.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Configuring Receivers

- Use the `createReceiverConfiguration` method to configure a receiver configuration for a receiving organization—by creating a `ReceiverConfiguration` value object.
- Use the `removeReceiverConfiguration` method to remove a receiver configuration (`ReceiverConfiguration` value object) for a receiving organization.
- Use the `updateReceiverConfiguration` method to update a receiver configuration for a receiving organization (updates a `ReceiverConfiguration` value object).

Configuring Triggers

- Use the `createTriggerConfiguration` method to configure a trigger configuration for a receiving organization—by creating a `ReceiverTriggerConfiguration` value object.

- Use the `removeTriggerConfiguration` method to remove a trigger configuration (`ReceiverTriggerConfiguration` value object) for a receiving organization.
- Use the `updateTriggerConfiguration` method to update a trigger configuration for a receiving organization (updates `ReceiverTriggerConfiguration` value object).

4.13.3.5 Configuring Receiver Vocabulary

OMP has leveraged ETS functionality to let outbound messages contain coded elements using a different `codingSystem` than the `codingSystem` value stored in HTB. Perform the following steps to facilitate transmission of outbound coded elements:

- Add coding system to ETS; if a new coding system, it must be added to ETS.
- Add code to ETS; if these are new codes, they must be loaded as part of a `codingSystem`.
- Add `conceptLists` to ETS; add codes from the coding system to custom `conceptLists`.
- Select attributes to elaborate and which `conceptLists` to use; select those coded attributes from the CTB_OMP_ATTRIBUTE_NAME `conceptList` that are to be elaborated.

Reference

Oracle Javadoc for HTB

Table 4–106 lists the principal OMP Receiver Vocabulary Service and methods:

Table 4–106 Service and Methods: OMP Receiver Vocabulary Configuration

Level	Detail
Package	`Oracle.apps.ctb.message.configuration`
Class	`OMPConfigurationAdministrationService`
Methods	▪ `createVocabulary` ▪ `updateVocabulary` ▪ `removeVocabulary`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the responsibility created in Section 4.13.3.1.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Configuring Vocabulary

- Use the `createVocabulary` method to create a receiver vocabulary configuration for a receiving organization—by creating a `ReceiverVocabularyConfiguration` value object.
- Use the `removeVocabulary` method to remove a vocabulary configuration (`ReceiverVocabularyConfiguration` value object) for a receiving organization.
- Use the `updateVocabulary` method to update a vocabulary configuration for a receiving organization (updates a `ReceiverVocabularyConfiguration` value object).

4.13.3.6 Configuring Clinical Concepts

Set up a Microsoft Excel file to load the clinical concepts configuration and save in .csv format. The file should contain the following fields:

- `ClinicalCategoryCode`
- `CMETID`
- `MessageTypeCode`, with the following values:
  - `EncEvn` for Encounter Event/Patient Admin messages.
  - `SA_ORDERS` for the Substance Admin Order messages.
  - `Scheduling` for the Scheduling/Encounter Appointment messages.

Table 4–107 illustrates sample data for such a file:

Table 4–107 Clinical Concepts File: Sample Data

Clinical_Category_Code	CMETID	Message_Type_Code
ADMDX	COCT_MT120100HT02	EncEvn
DISDX	COCT_MT120100HT02	EncEvn
INTDX	COCT_MT120100HT02	EncEvn
EXPECT_SURG	COCT_MT120100HT02	EncEvn
EXPECT_DSCH	COCT_MT120100HT02	EncEvn
TRANSF_REASON	COCT_MT120100HT02	EncEvn
ADMT_REASON	COCT_MT120100HT02	EncEvn
DIET_PREF	COCT_MT120100HT02	EncEvn
REL_CLIN_INFO	COCT_MT120100HT02	EncEvn
REASON	COCT_MT120100HT02	EncEvn
ACT_REASON	COCT_MT120100HT02	EncEvn
ADMDX	COCT_MT120100HT02	SA_ORDERS
DISDX	COCT_MT120100HT02	SA_ORDERS
INTDX	COCT_MT120100HT02	SA_ORDERS
REASON	COCT_MT120100HT02	SA_ORDERS
ACT_REASON	COCT_MT120100HT02	SA_ORDERS
NOTE	COCT_MT120100HT02	SA_ORDERS
REASON	COCT_MT120100HT02	Scheduling
ACT_REASON	COCT_MT120100HT02	Scheduling
ADMDX	COCT_MT120100HT02	Scheduling
DISDX	COCT_MT120100HT02	Scheduling
INTDX	COCT_MT120100HT02	Scheduling
NOTE	COCT_MT120100HT02	Scheduling

Use the following commands to load the .csv file into HTB:

Usage:

```
    java -classpath <CLASSPATH> -DCsvFileName="<CSV_FILE>" -
DDbUrl="<CONNECT_STRING>"
    -DUser=">APPS_USER>" -DPassword="<APPS_PWD>" -DIMP_DEBUG="Y"
    oracle.apps.ctb.message.omprocessor.util.ClinicalConceptSeeder
```

Example 4–23 Code Sample:

```
java -classpath ,:$CLASSPATH -DCsvFileName="ClinConcept.csv"
    -DDbUrl="jdbc:oracle:thin:@ap950sun.us.oracle.com:1521:hcxdr2"
    -DUser="apps" -DPassword="apps" -DIMP_DEBUG="Y"
    oracle.apps.ctb.message.omprocessor.util.ClinicalConceptSeeder
```

Note: Please ensure that the following files are part of the `CLASSPATH`:

- `htbclint.jar`
- `classes12.jar`
- `xmlparserv2.jar`

4.13.3.7 Configuring Oracle Advanced Queuing

Steps

1. Create an HTBGateway database user (schema) called *HTBGWY (HTBGWY schema*). The Database Administrator should make the following grants to this schema:
   - CONNECT
   - RESOURCE
   - create session
   - `aq_administrator_role`, with admin option
   - execute on `dbms_adadm`
   - execute on `dbms_aq`
   - execute on `SYS.AQ$_JMS_TEXT_MESSAGE`, with grant option to ctb
2. Execute the SQL script `$CTB_TOP/patch/115/sql/CTBCRJMS.sql` to create each of the receiver organizations. You are prompted for schema name, schema password, queue names and HTBGWY schema name, where the queue name should be what is specified as part of the `ReceiverConfiguration.setQueueLocationName` method.

Example 4–24 Removing Queue Name

To remove the Queue name `CTB_TEST_MS_RX1_ADT`:

```
sqlplus htbgwy/htbgwy @$CTB_TOP/patch/115/sql/CTBRMJMS.sql
ctb <ctb schema pwd> CTB_TEST_MS_RX1_ADT
```

Example 4–25 Creating Queue Name

To create the Queue name `CTB_TEST_MS_RX1_ADT`:

```
sqlplus htbgwy/htbgwy
@$CTB_TOP/patch/115/sql/CTBCRJMS.sql ctb <ctb schema pwd>
CTB_TEST_MS_Rx1_ADT htbgwy 100 0 0
```
(refer to the script for details about the parameters description)

3. Create a synonym for `ctb.ctb_ms_receiver_config` in the HTBGWY schema as `ctb_ms_receiver_config`.
4. Grant select and update privileges to `ctb.ctb_ms_receiver_config` to the HTBGWY schema.

Note: All required scripts are provided with HTB.

See Also:

- *Oracle Advance Queuing Application Developer's Guide*
- *Oracle9i supplied PL/SQL Packages and Types Reference*

4.13.3.8 Configuring Oracle Workflow

An application based on the HTB Platform will be able to initiate a request to generate and send an outbound message which is raising a business event. Each business event is configured within Oracle Workflow to have one or more workflow processes subscribed to it.

When the `OMPService` methods are called, it raises a particular business event based on the trigger event passed, which executes the subscribed workflow processes.

HTB is delivered with a preconfigured workflow process that can be used to generate and send outbound messages—based on standardized assumptions regarding the processing sequence and error handling procedures. The preconfigured workflow process is automatically loaded during the installation

(located at $CTB_TOP/patch/115/import/US ctbomp.wft on the APPL_TOP where the product is installed).

Confirm that the Workflow Administrator Web Applications responsibility is assigned to the Workflow Administrator, by selecting the Global Workflow Preference option on the Self-Service Application. If it is not, request the System Administrator to assign the Workflow Administrator Web Applications responsibility to the Workflow Administrator.

> **See Also:** Appendix G, Seeded OMP Workflow Data, for information about seeded workflow data, including workflow processes, business events, and event subscriptions for HTB Outbound Messaging Services.

You can reconfigure these workflow processes by accessing Oracle Workflow:

- After the preconfigured workflow is loaded into the database (as part of the installation), two attributes must be modified in the workflow process.
- Using *Oracle Workflow Builder 2.6.3* with access level set to 20, select and open the CTB OMP by connecting to the database.
- Expand CTBOMP and expand CTBOMP Item Type (click the + sign in the Navigator window).
- Expand the attributes (click the + sign in the Navigator window).
- Double-click the User Id attribute and set the value to OMP Retrieval User Account in the Attribute tab.
- Similarly double-click the User Credential attribute and set the value to the password of the OMP Retrieval Account in the Attribute tab.
- Also double-click the Purpose Code attribute and set the value to one of the purpose code values that belong to the concept list CTB_SEC_PURP.
- Save the workflow process to the database.

> **See Also:**
>
> - *Oracle Workflow Administrator's Guide, Release 11i*
> - *Oracle Workflow Developer's Guide, Release 11i*
> - *Oracle Workflow Builder, Release 11i*

4.13.3.9 Configuring Business Event System

All the Oracle workflow related Business Event System metadata is automatically loaded during installation.

> **See Also:** Appendix G, Seeded OMP Workflow Data, for information about seeded workflow data, including workflow processes, business events, and event subscriptions for HTB Outbound Messaging Services.

4.13.3.10 Setting up HTB Gateway

- Set up the HTB Gateway and configure to the external interface engine. Copy the *HTBGateway.zip* file from *$CTB_TOP/java* and unzip it into a directory named *HTBGATEWAY_HOME* (unzips into *HTBGateway_D*).

  See Also:

  - Oracle Healthcare HTB Gateway Installation Configuration (HTBGatewayInstall.doc) for more information about configuring the HTB Gateway; this document is stored under the *$HTBGATEWAY_HOME* directory.
  - *DOWNLOAD_README.TXT,* contained in the *HTBGateway.zip* file, for instructions about how to download the open source libraries.

- Make modifications to *HTBGateway_D/config/config.xml* to point to the correct hostname/port and database configuration—similar to *HTBGateway_D*. Required changes follow:
  - Change the `<APPS_SCHEMA>` to point to the machine name where the database instance is hosted.
  - Change the `<QUEUE_SCHEMA>` to point to the machine name where queues are created (the `QUEUE_SCHEMA` points to the machine where the database is hosted).

Example 4–26 Sample Script (HTBGateway)

```
<HTB_REPOSITORY_INFO>
<APPS_SCHEMA>
<HOST>ap049sun.us.orcle.com</HOST>
<PORT>19701</PORT>
<SID>hcxd2rd</SID>
<USERNAME>htbgwy</USERNAME>
```

```
<PASSWORD>htbgwy</PASSWORD>
</APPS_SCHEMA>
<QUEUE_SCHEMA>
<HOST>1p049sun.us.oracle.com</HOST>
<PORT>19701</PORT>
<SID>hcxd2rd</SID>
<USERNAME>htbgwy</USERNAME>
<PASSWORD>htbgwy</PASSWORD>
<QUEUE_OWNER>ctb</QUEUE_OWNER>
</QUEUE_SCHEMA>
<APPS>
    <IDENTITY><OMP Retrieval User Account as set above></IDENTITY>
    <CREDENTIAL><OMP Retrieval User Account Credential (password) as set
    above></CREDENTIAL>
    <RESPONSIBILITY></RESPONSIBILITY>
    <LOGIN_ORG_ID></LOGIN_ORG_ID.
    <PURPOSE_CODE></PURPOSE_CODE>
</APPS>
</HTB_REPOSITORY_INFO>
```

– Configure the receiving system. All of the entries for the receiving system should be changed for `hostname`:

```
<RECEIVING_SYSTEM_CHANNELS DESC="2.1.2.2">
<MLLP_CHANNEL DESC="2.1.2.2">
    <HOSTNAME>hostname.domain.com</HOSTNAME>
    <PORT>7004</PORT>
    <CONN_RETRIES>5</CONN_RETRIES>
    <SEND_RETRIES>5</SEND_RETRIES>

<PAUSE_TIME_BETWEEN_CONNECTS>5</PAUSE_TIME_BETWEEN_CONNECTS>
    <RECEIVE_TIMEOUT>5</RECEIVE_TIMEOUT>
    <SEND_TIMEOUT>5</SEND_TIMEOUT>
    </MLLP_CHANNEL>
</RECEIVING_SYSTEM_CHANNELS>
```

– Run the following SQL statement to see if aq_tm processes > 1:

```
select name,value from v$parameter where name='am_tm_processes';
If not, change the init.ora parameter to greater than 1 and bounce the
database.
```

Steps

Set up a configuration file (call it `config.xml`):

1. Set up the MLLP ports, including the filters to be used, filter sequence and the filter parameters.
2. Set up the HTTP ports for gateway administration.
3. Set up the security password and timeout.
4. Set up the log file.
5. Set up the repository information, including database and queue schema.
6. Set up the receiving system configuration (external interface engine), including the port and host it is listening on.
7. Start the gateway ($HTB_GATEWAY_HOME/start.sh).
8. Stop the gateway (see: HTB Gateway Command Line Options).

HTB Gateway Command Line Options

Confirm that `$HTBGATEWAY_HOME/lib/htbgateway.jar` is in the CLASSPATH:

- To get the usage:
  ```
  java
  oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin/?
  ```
- Usage:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  [-createpass][/?][-pass <pass>
  [start|stop|startall|stopall|ch_starttall|ch_stopall|ch_
  quitall|ch_activateall|ch_deactivateall|ch_start_rec|ch_
  stop_rec|ch_activate_rec|ch_deactivate_rec|ch_start_mc|ch_
  stop_mc|ch_activate_mc|ch_deactivate_
  mc][adminhostname1:adminport1,adminhostname2:adminport2][[p
  ortname]|[receivername]|[receivername][channelname]]
  ```
- To create password:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -createpass <password>
  ```
  You will get an encrypted password that you must copy to config.xml without which the subsequent commands cannot be executed.
- To start all inbound MLLP ports:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> startall <adminhostname:adminport>
  ```

- To stop all inbound MLLP ports:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> stopall <adminhostname:adminport>
  ```
- To start an inbound MLLP port:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> start <adminhostname:adminport><portname>
  ```
- To stop an inbound MLLP port:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> stop <adminhostname:adminport><portname>
  ```
- To start all outbound MLLP channels:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_startall <adminhostname:adminport>
  ```
- To activate all outbound MLLP channels:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_activateall <adminhostname:adminport>
  ```
- To deactivate all outbound MLLP channels:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_deactivateall <adminhostname:adminport>
  ```
- To stop all outbound MLLP channels:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_stopall <adminhostname:adminport>
  ```
- To abort all outbound MLLP channels:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_quitall <adminhostname:adminport>
  ```
- To start all outbound MLLP channels for a receiver:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_start_rec
  <adminhostname:adminport><receivername>
  ```
- To activate all outbound MLLP channels for a receiver:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_activate_rec
  <adminhostname:adminport><receivername>
  ```
- To deactivate all outbound MLLP channels for a receiver:
  ```
  java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
  -pass <password> ch_deactivate_rec
  <adminhostname:adminport><receivername>
  ```

- To stop all outbound MLLP channels for a receiver:
```
java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
-pass <password> ch_stop_rec
<adminhostname:adminport><receivername>
```
- To start outbound MLLP channel:
```
java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
-pass <password> ch_start_rec
<adminhostname:adminport><channelname>
```
- To activate outbound MLLP channel:
```
java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
-pass <password> ch_activate_rec
<adminhostname:adminport><channelname>
```
- To deactivate outbound MLLP channel:
```
java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
-pass <password> ch_deactivate_rec
<adminhostname:adminport><channelname>
```
- To stop outbound MLLP channel:
```
java oracle.apps.ctb.message.mdi.htbgateway.HTBGatewayAdmin
-pass <password> ch_stop_rec
<adminhostname:adminport><channelname>
```

See Also: *Oracle Healthcare HTB Gateway Installation Configuration* (`HTBGatewayInstall.doc`) for more information about configuring the HTB Gateway. This document is stored under the *$HTBGATEWAY_HOME* directory.

Check Points:

- Confirm that you have access to the Oracle Application Manager URL (provided by the System Administrator) and log in as System Administrator.
- Confirm that the Oracle Concurrent Manager and Oracle Workflow listeners *(Workflow Java Deferred Agent Listener, Workflow Java Error Agent Listener)* are up and running.
- Contact the System Administrator for further details.

See Also:

- *Oracle Workflow Administrator's Guide, Release 11i*
- *Oracle Workflow Developer's Guide, Release 11i*
- *Oracle Workflow Builder, Release 11i*

4.13.4 Invoking Outbound Messaging Services

OMP services support the following core functions:

- Generating Outbound Messages
- Processing Message Acknowledgements
- Resending Messages
- Skipping Messages
- Error Logging

Reference

Oracle Javadoc for HTB

Table 4–108 lists the principal OMP service and methods:

Table 4–108 Service and Methods: OMP

Level	Detail
`Package`	`oracle.apps.ctb.message.omprocessor`
`Class`	`OMPService`
`Methods`	▪ `acknowledgeMessage` ▪ `generateMessage` ▪ `resendMessage` ▪ `skipMessage`
`Class`	▪ `ErrorLogService`
`Methods`	▪ `createError` ▪ `getErrorByMessageId` ▪ `updateError`

Login

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

Responsibility

Use the seeded Outbound Message Processor Responsibility.

Navigation

This is an API-based implementation procedure.

See: Appendix A, HTB Session Service

4.13.4.1 Message Formulation and Transmission

Use the following `OMPService` methods to generate and process outbound messages:

- Use the `generateMessage` method to trigger the generation of an outbound message. The OMP Message Generator extracts all necessary value object attributes, formulates an HL7 version 3 conformant message, logs it to the HTB Repository, and queues it for sending—in accordance with processes configured (for each receiving organization) in Oracle Workflow for each receiving organization. The message is actually sent (to the interface engine) upon polling of the outbound queue by the HTB Gateway, in accordance with workflow constraints.
- Use the `acknowledgeMessage` method to process incoming message acknowledgements from receiving systems—triggered by receipt of an outbound HL7 message sent by OMP. There are three possible message acknowledgement codes, but only two are supported by OMP: (i) Application Acknowledgement (AA), and (ii) Application Error (AE).
  - An AA response indicates that the message was successfully transmitted and received by the receiving application. OMP logs the response and attaches it to the original message, closing it out.
  - An AE response indicates an error reported by the target application, including error information in functional segments (includes an error type code, error detail code and a free text message). OMP logs the error response, attaches it to the original message, and initiates a workflow notification, which calls error processing. OMP continues to suspense the message until the error condition is resolved, which may require human

intervention. Resolution means correcting and resending the message (`resendMessage`) or skipping it (`skipMessage`).

- Use the `resendMessage` method to resend a corrected message to the original target following an AE acknowledgement from the target application. Correction of the original message requires manual intervention by an authorized user (OMP Administrator or other user with *Resend* function attached) to query the old message, change it, and resubmit the modified message.
- Use the `skipMessage` method to resolve a message with an AE acknowledgement by marking it to be skipped—which effectively cancels the message. Must be initiated by an authorized user (OMP Administrator or other user with *Skip* function attached).

4.13.4.2 Error Logging

Use the following `ErrorLogService` methods to process OMP Application Errors:

- The `createError` method is used by OMP services to log message errors to the HTB Repository. Applies to outbound messages with an AE acknowledgement as well as to message requests that fail during processing. OMP first uses `createError` to log any errors that occur during message generation. OMP also logs the error response received from an acknowledgement message, and attaches it to the original message. This closes out the original message, terminating it from the outbound queue.
- Use the `getErrorByMessageId` method to retrieve any error text associated with a message by its `messageId`.
- Use the `updateError` method to update the `Error.resolved` flag to indicate that this error has been resolved (defaults to *F*). Use `getErrorByMessageId` to retrieve the error text information.

4.13.5 OMP Security

OMP generates and sends outbound messages to both internal and external systems—that may contain **protected health information**. Accordingly, security service calls are imbedded in all API calls (throughout HTB) to query, create, and update patient information. OMP APIs are trusted when interacting with other HTB APIs, and protect against redundant security calls by calling the HTB Patient Consent Service directly. OMP calls the following services to implement security:

- **Message Generator:** Called by HTB applications to generate and send outbound messages. Because such messages may include protected health

information, OMP calls the Consent Service directly to ensure that appropriate patient authorization has been granted. If such authorization does not exist, generation of the outbound message is inhibited and OMP logs an error.

- **Receive Acknowledgement:** Called by HTB applications to process message acknowledgements. Because this function does not interact with a user, and does not send patient information to another system, it does not check for patient authorization.
- **Skip Message:** Called by HTB applications to skip retransmission of a message marked with an Application Error acknowledgement (AE). Because this function does not send patient information to another system (it inhibits retransmission), it does not check for patient authorization.
- **Resend Message:** Called by HTB applications to resend a manually modified message to another system. Although functional security is implemented for this call, OMP does not reverify patient authorization.

See Also:

- Section 4.1, Implementing Security Services
- Section 4.7, Implementing Consent Services

4.13.6 OMP Audit

The HTB Audit Service lets you log HTB activities and record which APIs a particular user or process has called during a session.

OMP directly calls the HTB Audit Service to log the following events:

- Generate Message
- Skip Message
- Resend Message

See Also: Section 4.6, Implementing Audit Services

5

Implementation Tasks: Payer Financial Management

This chapter provides an overview of Payer Financial Management (PFM), a component of Oracle Healthcare Transaction Base (HTB), including implementation procedures and PFM report descriptions. It also provides system implementors with a high-level guide for the design and development of a customized extraction, transformation, and load process (ETL), which is required to transfer data from source transactional systems into the PFM financial repository.

This chapter contains the following topics:

See Also:

- Chapter 3, Implementation Overview, and
- Chapter 4, Implementation Tasks: HTB Platform,

...for information about implementing the Oracle Healthcare Transaction Base Platform.

5.1 Payer Financial Management Overview

PFM is a claims-centric financial transactions repository. Payers can use PFM to persist financial transactions generated by a claims processing system into a single repository that is centered on healthcare claims and other user-defined agency transaction types. The repository in turn supports reporting of historical financial transactions for single or multiple claims. It also supports summary financial reporting by claim.

PFM lets payers analyze incoming payable claims in the context of financial events that take place during payable claims processing. This system provides both open interface tables and an Applications Programming Interface (API). Using this interface, users can populate core tables with transactional data from existing claims processing systems and Oracle Financial modules, and query the repository as well. API queries can retrieve claims and other key transactions by multiple dimensions, such as providers or accounting periods. Users can analyze claims as they relate to additional events, such as payable invoices, adjustments, and payments.

In addition to producing claims and other key transactions associated with events, the PFM can also report summary accounting entries associated with financial events related to a claim at the claim level, providing a quick summary view of the accounting impact of a claim.

By using an open interface approach, PFM can accommodate claims data from a variety of existing payer claims processors. As part of *Oracle E-Business Suite*, PFM works with *Oracle Financials* (AP, AR, and GL) to let users analyze the financial impact of claims.

5.1.1 HTB Claims-centric Repository – Financial Data

This repository stores a claim as a central document called an *agency transaction.* Financial events and data are clustered around the agency transaction for business intelligence and decision support purposes—and can be queried by API calls. This repository lets users summarize claim-related financial transactions at the claim level. It also lets users view detailed financial transactions related to a claim, through detailed drill-down queries.

In addition to being claims-centric, the repository is *agency transaction-centric*—various classes of financial transactions termed an agency transaction serve as a summary point for all related financial events.

5.1.2 Agency Transactions

In addition to claims, PFM is *extensible*—agency transactions **other** than claims are supported as a key document around which events are centered. Users can use the same APIs and queries on non-claim documents to summarize and report financial and custom events. These other agency transactions include:

- Manual AP Invoices
- Manual AR Invoices
- MSP (Medicare Secondary Payer) Invoices
- Prepayments
- No-Pay Claims
- Unapplied Receipts
- General Ledger Journal Entries

These agency transactions are supported by repository APIs, queries and claims

5.1.3 Events

Events are the financial and non-financial transactions from source systems associated with a claim or another type of agency transaction. These events are imported from the source system into PFM though open interface tables (provided through *Oracle eTRM for HTB*). Some of the types of financial events associated with claims include the following:

- Payable Invoices that result from a claim
- Adjustment claims
- Claim invoice payments
- Claim invoice payment clearing

PFM supports all financial events captured by *Oracle E-Business Suite Financial Applications.* PFM also supports custom, user defined events. Examples of such custom events include the following:

- AR Invoice status changes
- Dunning letters sent on late invoices

> **See Also:** Section 5.5, PFM Open Interface Tables, for more information about open interface tables.

5.1.4 HTB Transaction-centric Open Interface

An open interface table lets you populate the HTB repository with financial and non-financial events. Financial events are part of *Oracle E-Business Suite* functionality. Users capture selected E-Business Suite and customized events, and pass them to HTB using the interface table.

In addition to being the primary mechanism for migrating events into HTB, the interface tables also let the user's interface process tell HTB how such events are related to each other. Once the relationship between events and agency transactions are defined in the interface table and populated into HTB, HTB query APIs can report the appropriate history of events for each agency transaction.

> **See Also:** Section 5.5, PFM Open Interface Tables, for more information about open interface tables.

5.1.5 Standard Prototype Report

PFM includes a standard prototype report. This is a generic report that customers can customize to meet their specific requirements. The report shows the agency transaction and its related events. It also summarizes accounting entry amounts of those events at the agency transaction level. Using this report, customers can determine the current state of the agency transaction and the accounting impact of related events.

5.1.6 HTB Claims Access Query APIs

In addition to the standard query report, a set of query APIs is provided, letting users query claims and other agency transactions by a variety of dimensions, including the following:

- Agency transaction type
- Claim status (Open or Closed)
- Organization
- Provider
- General ledger accounting period

Users can imbed these APIs into their custom applications to obtain summary or detailed reports. For example, a payer may create a customer service application that calls the query API `OpenClaimsbyProvider` to monitor the number of unpaid claims for a specific provider. Alternatively, the application could also call

the `PaymentsbyProvider` query API to answer questions from providers about the status of their claim payments.

5.1.7 HTB Financial Audit Query APIs

Financial audit APIs let users perform financial audits within HTB, by producing financial distributions summarized by General Ledger accounts. Users can reconcile the sum total of historical transactions stored in HTB with General Ledger account balances from matching accounting periods. HTB also maintains traceability of financial events from HTB back to the source system, so that an auditor can trace a journal entry amount in Oracle General Ledger back to the source system (AP or AR), and from there back to HTB. Financial audit APIs include the following:

- Agency transactions by account by accounting period
- Financial event type by accounting period
- Transaction summary to detail reconciliation

Users can employ the first two APIs to reconcile from HTB to General Ledger account balances. The third API helps to ensure that within the Detailed Document Summary Report the events reconcile to the summary accounting balances at the agency transaction level.

5.2 Implementing Payer Financial Management

As part of *Oracle E-Business Suite,* PFM shares many common implementation steps with other Oracle Applications, including HTB. The dependencies are primarily on E-Business Suite Financials and on some HTB shared services, such as Enterprise Terminology Services (ETS).

As an implementor, you must first determine the business rules that determine how the transactions from your source system should appear on reports produced from this claims-centric repository. You must then create a custom program that extracts the claims-related transactions from the source system. Apply your own business logic to determine (i) which of those claims-related transactions is the original event that creates the agency transaction, and (ii) which serves as the anchor master record for all other subsequent claims-related events.

The next step is to define additional transactions to be extracted from source systems and transformed into the correct event types. You must clearly determine the links between events, agency transactions, summary distributions and other events. This information populates the PFM open interface tables.

Summary Distribution entries for each import batch must be created by the ETL process and inserted into the appropriate open interface table. The summary distribution definitions created during the prior implementation steps can be used to determine how the summary distributions are to be updated.

5.2.1 Implementing Oracle E-Business Suite Financials

From a payer's perspective, the results of claims processing are primarily accounts payable transactions. The payer issues payments to providers using a financial accounting system. As part of *Oracle E-Business Suite*, PFM supports transactions from the *Oracle Accounts Payable*. Claims processing may result in a receivable being created, where the payer is owed money from the provider. In this case, the payer must issue an accounts receivable invoice to the provider. For these transactions, PFM supports the transactions in the *Oracle Accounts Receivable* module.

In order to ensure that transactions from both modules can be reconciled and are complete from an accounting perspective, these AP and AR transactions must be posted to *Oracle General Ledger.*

See Also:

- *Oracle General Ledger User's Guide, Release 11i*
- *Oracle Accounts Payable User's Guide, Release 11i*
- *Oracle Accounts Receivable User's Guide, Release 11i*
- *Oracle Cash Management User's Guide, Release 11i*

...for information about implementing Oracle Financials.

5.2.2 Implementing Oracle Healthcare Transaction Base

You must implement the Oracle Healthcare Transaction Base Platform before implementing PFM.

See Also: Chapter 4, Implementation Tasks: HTB Platform, for more information about implementing the HTB.

5.2.3 Defining Organizations

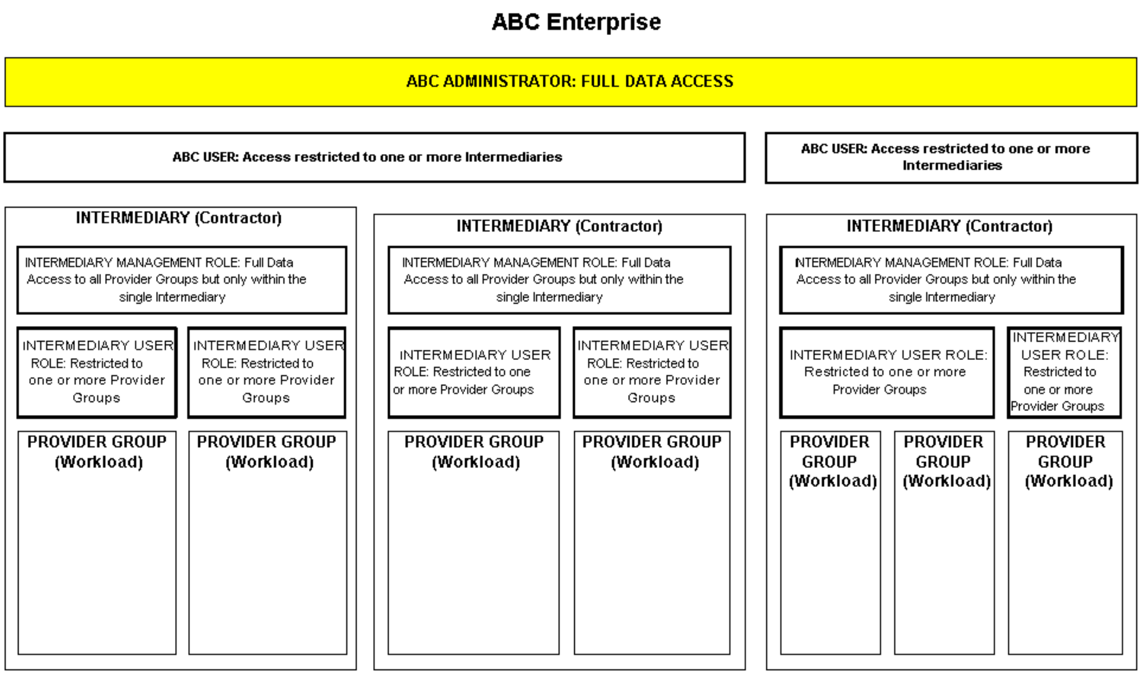
The organization is a key attribute that PFM uses for both query parameters and data security. Data security uses Organization ID as a securing attribute. Due to this dependency, the implementation of an organization hierarchy that properly reflects the true nature of the payer organization is required.

> **See Also:** Section 4.3, Implementing Organizations, for more information about organizations and organization hierarchies.

5.2.4 Implementing Security

PFM uses the Organization ID as the key attribute for securing access to data. Users assigned to one organization cannot query or view data that belongs to another organization. The security is hierarchical in nature. Users at a higher level of the organization hierarchy can view data across lower organization levels. Figure 5–1 illustrates the security domain:

Figure 5–1 PFM Security Domain



There are two levels of users: *Administrators* and *Users*. These two levels of users can be assigned to different organization levels, as shown in Figure 5–1.

It is important to note that the implementation team should define both the organizational setups and the security setups in combination, with careful consideration of the security requirements.

Security is implemented in the standard HTB and E-Business Suite methods. Users can be defined and assigned to responsibilities (Administrator and User level responsibilities), which are associated with the defined organizations.

> **See Also:** Section 4.1, Implementing Security Services for further information about Implementing HTB security.

5.2.5 Defining Summary Distributions

Consider agency transactions as master records. Transactions persisted from source systems as events under agency transactions are detail records. The summary distributions are the accounting entries of the financial events summarized at the master agency transaction record.

Summary distributions are grouped by the Oracle General Ledger account balancing segment and by the customer or supplier associated with the agency transaction in Oracle Accounts Receivable and Accounts Payable. This grouping logic is provided by default with PFM.

The accounting distributions of events can also be grouped by the natural account segment of the Oracle General Ledger code combination. A specific value in the natural account segment defines a specific type of account. For example, the value 3200 in the natural account segment may indicate that the money recorded in this account represents an Accounts Payable balance.

All event distribution rows contain a natural account segment. Each of these segments contains a unique value specifying whether that account represents Accounts Payable, Accounts Receivable, expenses, disbursements, collections, disbursements-in-transit, reconciled payments, or others. All financial classifications allowed by Generally Accepted Accounting Principles (GAAP) can be represented using a specific value in the natural account segment.

PFM lets you group your accounting distributions by these natural account segment values. For example, within an agency transaction, you can group all the accounting transactions balances by Accounts Payable, Accounts Receivable, and expenses. This enables you to view at a glance the total balance of all account classes associated with this agency transaction (or claim). You can define up to 30 different summary distribution groupings.

To define summary distribution groupings, define the code and names of each summary distribution group. Identify if the summary distribution represents debit

or credit balances as positive or negative numbers. Finally, define if the summary distribution represents the Accounts Payable or Accounts Receivable balance. One of each is required at a minimum across all 30 summary distributions.

After defining the summary distribution groups, you can then create the natural account segment ranges that define these groups.

Summary Distributions are defined in the following tables, provided through *Oracle eTRM for HTB*—you can use PL/SQL routines to populate them at setup:

- **Summary Distributions Setups** [CTB_FI_SUMMARY_CLASSES]

  In the Document Summary Report, all accounting distributions for financial events are summarized by user-defined classifications. This table lets users define those summary classifications. Users are required to define the various categories into which the accounting distributions are summarized. A maximum of 30 summary distributions are supported.

- **Distribution Classification Rules** [CTB_FI_SUMMARY_MAPS]

  The CTB_FI_SUMMARY_CLASSES table defines how accounting distribution data is classified. This table maps the actual rules used by the system to perform such classifications. The accounting distributions are grouped by the natural account segment of the Oracle General Ledger accounting code combination key flexfield. In this table, users must define, for each summary distribution setup in the prior table, what the values of the natural account segment are that map to a particular summary distribution. The ranges must be defined by high and low values for the natural account range.

  See Also:

  - Section 5.5, PFM Open Interface Tables
  - *Oracle eTRM for HTB*, available through Oracle*MetaLink*

5.2.5.1 Summary Distribution Segment Ranges

Summary distribution segment ranges are the range of values of the natural account segments that you have defined in Oracle General Ledger. You can define multiple ranges for each summary distribution grouping. The ranges can be non-contiguous and can have a range of a single value, but the ranges cannot overlap. For example, you can define a grouping called *Accounts Payable*. For this summary distribution, the ranges of values could be:

- 3200 – 3300
- 3700 – 3750

- 2000 – 2100
- 4000 – 4000

However, inserting another range such as 3299 - 3350 is not allowed, because it overlaps with the first range of 3200 – 3300.

> **See Also:** Oracle General Ledger User's Guide, Release 11i, for further information about General Ledger code combinations, balancing and natural account segments, or other accounting issues.

5.2.6 Extending Enterprise Terminology Services Lookup Codes

HTB is delivered with a predefined set of concept lists and their associated membership codes. These lists support eight different types of agency transactions and the list of events to support the transactions that occur in Oracle Financials. If additional agency transaction types are required to represent all the required claims or master document types at the agency transaction level, or if additional seeded event types are required to support all of the events, you can then extend the lookup codes in the ETS Shared Service in HTB. Table 5–1 lists ETS concept lists that are extensible for PFM:

Table 5–1 ETS Concept Lists Extensible for PFM

Concept List Name	Purpose
CTB_FI_AT_TYPE_CODE	Used to define agency transactions.
CTB_FI_AT_TYPE_CODE	Used to define events.
CTB_FI_SOURCE_TABLE_CODE	Used as a source system table identifier.

> **See Also:** Section 4.5, Implementing Enterprise Terminology Services for detailed information on the ETS process.

5.3 Implementing the Extraction, Transformation and Load Process

PFM is dependent upon the implementation of an *Extraction, Transformation and Load process (ETL)* to extract events from Oracle Financials. This process also determines how the HTB open interface tables are populated.

Each implementation has unique requirements, requiring different ETL processes. Mapping of transactions from the source system is dependent upon the

requirements for the report output. Clear definition of report output requirements is thus a prerequisite to design of the ETL process.

This section provides guidelines for designing the ETL process. Implementors must analyze in detail the unique requirements of their own implementation and tailor their ETL processes accordingly.

This section focuses on the extraction and transformation portion of the ETL process. The loading portion, for the purposes of implementation, means the population of the interface tables. The actual loading of the data into core tables is addressed by the product itself.

Note: A fundamental assumption is that the claim, or an agency transaction can be identified by a unique ID easily accessible to the ETL process.

5.3.1 Key ETL Elements

The ETL process should reflect the following considerations:

- Capturing all accounting events and transactions that occur in the source systems (AP, AR, and GL) so that it is possible to reconcile from HTB to any one of these source modules.
- Considering extracting only events that have been posted to Oracle General Ledger to ensure reconciliation.
- Capturing data in the posted journal entries into Oracle General Ledger, and then using the GL_IMPORT_REFERENCES table to query the detailed transactions in the source systems.
- Capturing all non-financial or custom events necessary to meet the requirements.
- Capturing all the data required to populate all required tables in HTB.
- Mapping the data extracted from the source systems into the proper HTB event categories.
- Determining which HTB Interface table to populate.

5.3.2 About HTB Event Categories

PFM supports all the transactions and accounting events that occur in Oracle Financials. These are represented in PFM and on the document summary report as

certain event types, such as AP Invoice, AP Payment, CM Reconciliation, among others. This list of event types is extensible by the implementation team.

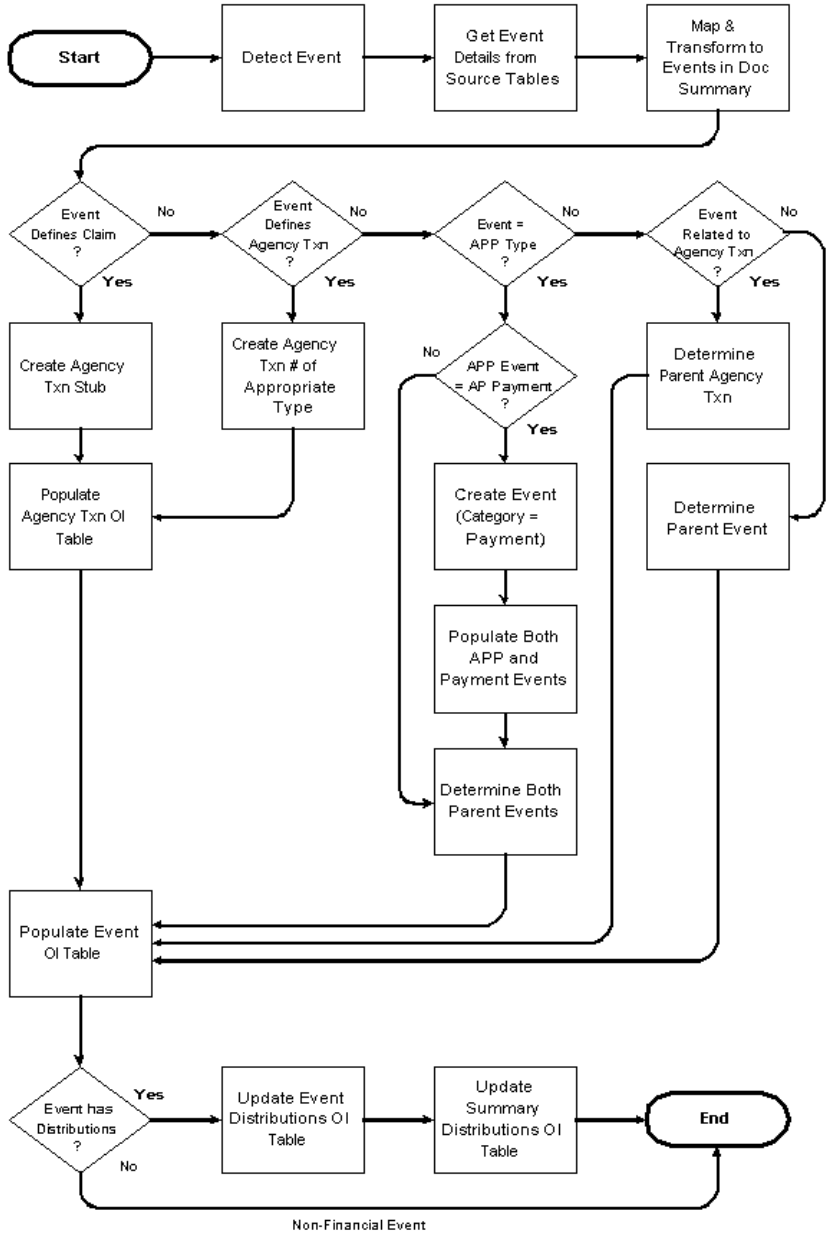
All event types, including custom-extended event types, have one of four event categories. The ETL process must map the extracted transactions and events from the source systems into one of these four categories. This lets the internal PFM logic determine the proper handling of these events. PFM supports the following event categories:

- **Financial Events:** Those that have accounting distributions lines. These events relate either directly to an agency transaction, or to other events. Agency transactions may be created because of the creation of a financial event. For example, an AP Invoice creation is a financial event, but if the AP Invoice being created is a claim invoice, a claim agency transaction is also created.
- **Non-financial Events:** Events that do not have accounting distribution. These events are also related to other events.
- **Payment Events:** These events represent a check in the AP system; they do not have accounting distributions.
- **Application Events:** These events represent the linkage between one event and another, where a many-to-many relationship may occur. For example, a check in AP can pay multiple AP Invoices, while one AP Invoice can be paid by many checks.

5.3.3 Model ETL Process

Figure 5–2 displays a model process flow for the ETL process. Modify this process as appropriate to the requirements of your own implementation:

Figure 5–2 Model ETL Process



The following sections describe the model ETL process illustrated by Figure 5–2:

1. **Start Process:** The ETL program is envisioned as a batch process that runs on a periodic basis. The determination of the frequency of the run is up to the end user. Accounts Receivable transactions should be posted to General Ledger prior to running this program.

2. **Detect Events:** This is the start of the extraction portion of the process. The process begins with posted journal entries in General Ledger. The process uses the record in the GL_IMPORT_REFERENCES table to point back to the specific transaction in sub-ledger source systems. From there, the logic should sweep through Oracle Accounts Payable, Accounts Receivable and General Ledger modules and search for the accounting distributions that have been posted to the General Ledger—that have not been extracted to HTB in the previous batch run (including manual GL journal entries with a status of *posted* as well). The details of this process vary slightly for each module and event.

   This portion of the process also detects custom events. The method of detection depends on the implementation of the custom event, since custom event may include non-financial events. Users may consider using the audit trail functionality in *Oracle E-Business Suite* to detect changes in selected tables, and then extract the custom events from the audit trail shadow tables. This approach may be best suited for the non-financial event category.

3. **Get Event Detail From Source Tables:** Once the process has detected the posted accounting distribution lines, the extraction process traces the distribution lines back to the transaction record and gets the actual details of the transaction (AP invoice number, type, status, and transaction keys.)

4. **Map And Transform Events:** ETL uses its internal logic and implementation data, available in the source systems, to determine which one of the four event categories an event belongs to. Additionally, ETL determines which HTB event type to map a transaction. For example, a new AP invoice creation is detected in step 2, and data for it is extracted. Using the data collected in Step 2, step 3 determines that this AP Invoice is categorized as a financial event and a manual AP Invoice event type.

5. **Decision: Does this event define a claim?** In some cases, if the event is defined as a event category of *financial* and event type of *AP Invoice*, this enables the creation of an agency transaction, or a claim.

   If the event does result in an agency transaction, the populate agency transaction path is followed and an agency transaction = claim is created.

6. **Decision: Does this event define an agency transaction?** If the event did not result in a claim, it may still define the creation of other agency transaction types. If it does create an agency transaction such as a Manual AP Invoice, or an Unapplied Receipt, the populate agency transaction path is followed.

   The following agency transactions are supported:

   - Claim
   - No-Pay claim
   - Manual AP Invoice
   - Manual AR Invoice
   - Prepayment Invoice
   - MSP Case
   - Unapplied Receipt
   - GL Journal Entry

7. **Decision: Is the event an application event and does it join two other events?** For example, does it have two parent events, one an invoice and the other a payment? If so, the application event path is followed.

8. **Does the event relate directly to an agency transaction?** For example, an AP or AR Invoice generally relates directly to an agency transaction type of *claim*. If it does, the parent agency transaction must be identified. If it does not relate directly to an agency transaction, it must be related to another event. In that case, the parent event must be identified.

9. **Create an Agency Transaction Record:** Create an agency transaction record using the invoice number as the agency transaction number. The Loader program makes the final determination if an agency transaction must be created; this only creates the stub agency transaction. The internal loader program determines if the transaction is created or not, in order to avoid possible duplication.

10. **Create an Agency Transaction Record:** Create an agency transaction record of the appropriate type. If the event defines another type of agency transaction, an agency transaction record of the appropriate type is created.

11. **Determine Application Event Category:** If the event belongs to the application event category, the process must further identify if this application event is an AP Payment event, where an AP Invoice was paid by a check. If so, the process creates a Payment Category event.

12. **Determine Parent Agency Transaction:** If the event is directly related to an agency transaction, ETL must identify the parent agency transaction.
13. **Determine Parent Event:** If the event is related to another event, the parent event must be identified.
14. **Create Payment Event Category:** If it is determined that the application event relates an AP Invoice to an AP check, the process is slightly different. All Application events have two parents because they join two existing parent transactions. In the case of an AP Payment, the application joins an AP Invoice with an AP Check. However, in Oracle AP the check creation and the application occur at the same time. Therefore, the process must create a representation of the AP Check as a Payment event category prior to the creation of the Application event.
15. **Create Single Payment Event:** A characteristic of the Payment category event is that it has no distributions, even though it is not a non-financial event (non-financial events also lack accounting distributions). Since multiple AP Payment applications can relate back to a single check, the ETL must create a single payment event for all the application events with a common check ID.
16. **Post Application and Payment Event:** Once the Payment category event is created, both it and all the application events relating this check to all the invoices must be posted to the open interface table.
17. **Determine Both Payment Events: ETL n**ow determines both parents of the application event. In the case of AP payment applications, one of the parents will be an AP Invoice, and the other will be the a single AP Payment event created for all the application events with the common check ID number.
18. **Populate Event OI Table:** This is the general events interface table. The events categories that go into this table are financial, non-financial, and application events.
19. **Determine if non-Financial Event:** This step determines if this event is a non-financial event. If the event does not have accounting distribution lines, it is a non-financial event, and the process terminates. If the event does have accounting distribution lines, the process proceeds with the summary distributions update.
20. **Populate Event Distributions OI Table:** If the event has accounting distributions, the event distributions open interface table must be populated.
21. **Populate Summary Distributions OI Table:** Once this is populated, the summary distributions table for the specific agency transaction that the financial event falls under must be populated—with the incremental change

that the new financial event distributions will cause to the existing summary distributions.

This completes the logical process for a single event. ETL can either terminate processing or loop back to the next event, based on implementation-specific requirements.

5.3.4 Transactional Reconciliation and Error Validations

Steps

To ensure data integrity, perform the following steps for populating the open interface tables and the core PFM tables:

1. Run your custom ETL process to create batches of records in the open interface table.
2. Build into your ETL process its own internal validation routines that conform to your business rules, as well as a process log file or report output that reports how many records were processed per table by batch. If these exception handling routines show that errors have occurred, terminate processing until the errors are resolved.
3. Run the HTB Loader Validation Process. This process performs another validation on the data in the open interface tables, using HTB's own business rules for ensuring data integrity. This process checks to ensure that basic relational integrity is maintained, as well as conformance with pre-seeded ETS lookup codes.
4. After running the Validation program, run the HTB Loader Error Report. This report shows all records that have failed the loader validation process. Do not proceed until all of the reported issues are resolved. If there are any errors reported, they will be primarily be due to errors in the source system, or in the custom ETL process.
5. After the Error Report has shown no errors, run the Loader Transfer program to transfer the transactions from the open interface table into the core tables. Ensure that the number of records processed in the Loader Program matches the number of records processed by ETL.

5.4 Reporting

5.4.1 Document Summary Reports

5.4.1.1 Document Summary Report

This report lets you review one or more agency transactions at the summary level. It shows all of the agency transactions queried, along with summary data for each agency transaction. It also shows the most recent details of the last transactions or events that have affected this agency transaction, and it shows you at-a-glance the financial position of this agency transaction—the total balance of Accounts Payable, Accounts Receivable, and other Accounting Balances directly related to the financial activities generated by this agency transaction.

If the displayed agency transaction type is *claim*, it shows the latest status of the transactions related to this claim, including the amount remaining to be paid on this claim (the Accounts Payable balance). It also shows other accounting balances related to this claim.

5.4.1.2 Document Detail Report

The Document Detail report is the detailed, companion version of the Document Summary Report. In addition to showing the same summary level information, this report also presents all the detailed events that are related to this agency transaction. It shows all the financial events and their accounting distributions, including all the debits and credits to the applicable General Ledger accounts. It also shows all the non-financial events that the custom ETL process has imported.

By running this report, you can view in detail all of the financial transactions that have occurred for a particular claim. Since this report pulls together data from both Accounts Payable and Accounts Receivable and presents it in a claims-centric format, it lets you see at-a-glance all financial transactions related to a particular claim.

As with all other Oracle standard reports, you can customize this and the Summary Report to fit your presentation needs using standard Oracle Tools. The data for these reports are available in views that can be easily used to create other similar claims-centric reports that you may require.

5.4.2 Reconciliation Report

The Document Detail report shows all the transaction details and their accounting distributions, as well as the summary distributions associated with the agency

transaction at the higher level. This report lets you reconcile the detailed event distributions to the summary distributions. It sums up all of the event distributions and compares them to the summary distributions. It also reports any discrepancies between the two and lets you troubleshoot the ETL process for updating the summary distribution interface tables.

5.4.3 Loader Validation Error Report

Run this report after the PFM system administrator has run the Loader Error Validation process.

After your ETL process has finished populating the open interface tables, the system administrator runs the processes for importing data from the open interface tables into the core PFM tables within HTB. To proof against errors, the system administrator should run the Loader Validation Process and review any errors reported. After an analysis of the error report the system administrator can run the Loader Transfer program to import the data from the open interface tables into the HTB core tables.

The error report provides a view of any errors, error codes, and the meaning of the errors. Error records remain in the open interface table until purged, or until the fixed rows are re-validated and acceptable for import.

You should set up the Error Report as part of a Concurrent Manager Request Set together with the Loader Validation program.

5.4.4 Running Reports

For details about setting up and running reports using Oracle Concurrent Manager, see the following reference:

See Also: ***Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests***

5.5 PFM Open Interface Tables

PFM provides a set of open interface tables (through *Oracle eTRM for HTB*) to facilitate the migration of financial and non-financial events from *Oracle E-Business Suite* and other source files into the HTB Repository. These tables also let the user's interface process tell HTB how such events and agency transaction are defined in the interface table and populated into HTB. The following open interface tables are supported:

- **Agency Transaction Interface** [CTB_FI_AGENCY_TRANS_IFC}

  This table accepts agency transactions and related information. The custom logic of your ETL process determines when an agency transaction should be created, based on an event that occurs within Oracle E-Business Suite. If an event leads to the creation of an agency transaction in accordance with your custom ETL logic, the newly created agency transaction record is passed to this open interface table. In addition, if your unique ETL logic determines that an event should be associated with a previously created agency transaction, a reference to that agency transaction is also inserted into this table.

- **Events Interface** [CTB_FI_EVENTS_IFC]

  This table accepts all event data that is extracted through your custom ETL process. This lets you associate events with their parents. All events of all categories are imported through this table.

- **Events Distribution Interface** [CTB_FI_EVENT_DISTS_IFC]

  This table accepts accounting distributions associated with financial events passed to the Event Interface Table. For each financial event there are two or more records in this table. You must insert all accounting distributions for all events into this table to ensure that HTB can reconcile with Oracle General Ledger.

- **Summary Distribution Interface** [CTB_FI_SUMMARY_DISTS_IFC]

  This table accepts accounting distribution summaries of all distributions passed to the Events Distribution Interface Table. You must summarize all of the accounting distributions within your ETL batch by balancing segment and by party. This summary occurs in the context of your ETL batch. The HTB loader uses this data to build summary accounting distributions for the entire agency transaction.

See Also: *Oracle eTRM for HTB*, available from Oracle *MetaLink*

Part III

Appendixes

This section contains the following Appendixes:

A

HTB Session Service

HTB Session Service lets users interact with the HTB platform, and provides a call interface to all supported interfaces. It lets users initiate authenticated sessions, while maintaining context information. The session service has the following built-in context attributes (Table A–1):

Table A–1 Session Service Attributes[1]

Attribute Name	Description	Value
`Break-the-glass`	Whether to bypass authorization	'Y' or 'N'
`Enterprise Org ID`	The enterprise where user logs in	A valid org ID
`Language ID`	For localization purpose	`CTB_LANGUAGE_CODE`
`LoginOrg ID`	The organization where users logs in	A valid org ID
`LoginPersonId`	Person identifier of the user	A valid person ID
`User`	The authenticated user object	User

[1] Note that users can also define their own session attributes.

To initiate a user session you must execute a Java program, coded in substantial conformance with Example A–1, and customized to the unique requirements of your installation:

Example A–1 Sample Java Code to Create a Session

```
import oracle.apps.ctb.fwk.serviceLocator.common.*;
import oracle.apps.ctb.security.common.*;
import oracle.apps.ctb.security.*;
import oracle.apps.hct.base.*;
import oracle.apps.hct.base.ETSService;
import oracle.apps.hct.base.ETSAdministrationService;
import oracle.apps.hct.base.client.ETSAdministrationServiceImpl;
import oracle.apps.hct.base.client.ETSServiceImpl;
import oracle.apps.hct.base.ETSService;

public class SessionConnect
{
  private ServiceLocator serviceLocator = null;
  private SessionService sessionService = null;

  public SessionConnect()
  {
  }

  private void setupSession() throws Exception
  {
    System.out.println("Entering setupSession()");
    System.setProperty("ClientMode","remote");
```

```
  serviceLocator = ServiceLocator.getInstance();
  System.out.println("Received ServiceLocator");
  //Insert username/password here.
  serviceLocator.login("sysadmin","sysadmin");

  sessionService = serviceLocator.getSessionService();
  System.out.println("Received sessionService");

  Responsibility[] resp = sessionService.getResponsibilitiesForCurrentUser();
  String myRespName = null;
  Responsibility sysadmin = null;

    for (int i = 0 ; i < resp.length; i++)
    {
      if(resp[i].getName().equals("CTB_SECURITY_ADMIN_KEY"))
      {
        sysadmin = resp[i];
        break;
      }
    }
  sessionService.setResponsibilityForCurrentSession(sysadmin);
  //Breaking the Glass

  SessionContext sessionContext = sessionService.getCTBSessionContext();
  sessionContext.setBreakTheGlass("broken");
  sessionService.updateSessionContext(sessionContext);
  System.out.println("Sucessfully Broke the glass");

}
// Remember to logout
private void closeSessions() throws Exception
{
  serviceLocator.logout();
  System.out.println("Sucessfully logged out");
  System.out.println();
}
/**
 *
 * @param args
 */
public static void main(String[] args)
{
  try
  {
    SessionConnect sessionAnswer = new SessionConnect();
```

```
      sessionAnswer.setupSession();
      sessionAnswer.closeSessions();
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }

  }
}
```

HTB Session Timeout and Account Lockout

You can configure session timeout and account lockout by setting E-Business suite profile options.

> **See Also:** Section 4.1.1, Configuring Password and Session Policies

B

ETS Supported Terminologies and Cross Maps

This Appendix contains the following tables, listing terminologies and cross maps supported by ETS:

- Table B–1 defines supported terminology or vocabulary standards.
- Table B–2 lists cross maps that have been validated for ETS. The mappings are available to HTB customers, subject to prior licensing requirements for source and target terminologies.

Table B–1 ETS Supported Terminologies

Coding Scheme Name	Versions	Description	Source	Core ETS Terminology?	Seeded in ETS?
ANSI ASC X12 Healthcare Provider	2002, 2003, 2004	HIPAA Provider Taxonomy	Washington Publishing Company[1]	No[2]	No
CPT4	2002, 2003, 2004	Current Procedural Terminology, Fourth Edition	ADPContext	Yes	No
FDB[3]	Monthly, beginning in 2004	US FirstDataBank (FDB) National Drug Data File Plus (NDDF)	FDB	Yes	No
HCPCS-2	2002, 2003, 2004	Healthcare Common Procedure Coding System Level II	CMS Public Use Files[1]	Yes	No
HL7 v3[4]	1.16, 2.01	HL7 RIM v3 coding schemes	HL7	No[1]	Yes
HTB Supplemental	various	HTB Supplemental terminology (insernal Oracle terminology)	Oracle Healthcare	No[1]	Yes
ICD-10	1999	International Statistical Classification of Diseases and Health-Related Problems	WHO	Yes	No
ICD-9-CM-DRG	2002, 2003, 2004	US Diagnosis Related Groups	CMS Public Use Files[1]	Yes	No
ICD-9-CM-MDC	2002, 2003, 2004	US Major Diagnostic Categories	CMS Public Use Files[1]	Yes	No
ICD-9-CM-V1	2002, 2003, 2004	International Classification of Diseases, Ninth Revision	ADPContext	Yes	No
ICD-9-CM-V3	2002, 2003, 2004	International Classification of Diseases, Ninth Revision	ADPContext	Yes	No
IETF RFC 1766[5,6]	March, 2003	Internet Task Force Request for Comments 1766	ISO[1]	No[2]	Yes
ISO 11073-10101	November, 2003	International Standards Organization 11073-10101 Point of Care Device Codes	ISO[1]	No[2]	No
ISO 3166-1 alpha2[3]	1/18/03, 11/19/03	International Standards Organization 3166-1, 2 character country codes	ISO[1]	No[2]	Yes
LOINC	2.05, 2.10	Logical Observation Identifiers Names and Codes	Regenstrief Institute	Yes	No
NUBC-UB92	5/11/02, 9/17/03	National Uniform Committee Uniform Bill-92	National Uniform Billing Committee[1]	No[2]	Yes
SNOMED CT	1/02,7/02, 1/03, 7/03,1/04	SNOMED Clinical Terms	College of American Pathologists	Yes	No

[1] As formatted for ETS by Apelon, Inc.
[2] Supported as generic terminology.
[3] A subset is loaded into ETS.
[4] Implemented as multiple generic coding schemes.
[5] Referenced by HL7 as an external vocabulary domain.
[6] IETF 1766 references ISO 639-1; a subset of ISO 639-1 codes is loaded into ETS.

Table B–2 ETS Supported Cross Maps

Mapping	Source
FDB Diseases to/from ICD-9-CM	FDB (as formatted by Apelon, Inc.)
HL7 routes and forms to/from FDB routes and forms	Apelon, Inc.
LOINC (2.10) to/from CPT-4 (2003)	Apelon, Inc.
SNOMED-CT (1/03) to/from CPT-4 (2003)	Apelon, Inc.
SNOMED-CT (1/03) to/from ICD-9-CM (2003)	College of American Pathologists (as formatted by Apelon, Inc.)
SNOMED-CT (1/03) to/from ICD-10 (1999)	College of American Pathologists (as formatted by Apelon, Inc.)

C

Seeded Users, Responsibilities

This Appendix lists predefined (seeded) users, responsibilities, menus, grants, and criteria included with the HTB Platform, in the following tables:

- Table C–1, Seeded Users
- Table C–2, Seeded Responsibilities
- Table C–3, Seeded Grants

Table C–1 Seeded Users

User Name	Description
SYSADMIN	Only user name seeded in HTB. This user name also provides access to all functions within the E-Business Suite.

Table C–2 Seeded Responsibilities

Responsibility Name	Description
Healthcare Application Developer	Creates and updates Profile Options. See: Section 4.4, Implementing Profile Option Services
Healthcare Configuration Administrator	Creates and updates Profile Option values, including those for Lookup and Audit Services. See: ■ Section 4.4, Implementing Profile Option Services ■ Section 4.6, Implementing Audit Services
Healthcare ETS Administrator	Implements Enterprise Terminology Services. See: Section 4.5, Implementing Enterprise Terminology Services
Healthcare Security Administrator	Manages HTB Security Service, such as managing users, responsibilities, and grants.
Inbound Message Processor	Used by Inbound Message Processor to process inbound messages.
Outbound Message Processor	Used by Outbound Message Processor to process outbound messages.
Trading Community Manager	Implementing Person Services (TCA Responsibility)

Table C–3 Seeded Grants

Grant Name	Responsibility Name	Function Name
Security Grant	**Healthcare Security Administrator**	■ Break The Glass ■ Security Management
IMP Grant	**Inbound Message Processor**	■ Process Inbound Message
OMP Grant	**Outbound Message Processor**	■ Resend Skip Message ■ Compose Message ■ Get Receiver Configuration ■ Get Trigger Configuration ■ Get Vocabulary Configuration ■ Get Trigger Detail

D

Seeded Profile Options

Table D–1 lists predefined (seeded) profile options and associated lookup codes included with the HTB platform, sorted by Profile Option Code:

Table D–1 Profile Options

Code	Name	Description	Value Type Code	Default Value[1]
ADDRESS_STATUS_ ATTRIBUTE	Address Status Custom Attribute	Identifies which CUSTOM_ ATTRIBUTEXXX is used for Identification Number.	TEXT	N
CTB_AU_ADD_ RESPONSIBILITY	CTB: Audit Add Responsibility	Audit events of this type represent attempts to create security responsibilities.	CTB_YES_NO	Y
CTB_AU_ADD_RULE	CTB: Audit Add Rule	Audit events of this type represent attempts to create security rules.	CTB_YES_NO	Y
CTB_AU_ADD_USER	CTB: Audit Add User	Audit events of this type represent attempts to create system or user accounts.	CTB_YES_NO	Y
CTB_AU_ALL_WHEN_BTG	CTB: Audit All When BTG	When set to YES and when user breaks the glass, all defined audit events are audited.	CTB_YES_NO	Y
CTB_AU_AUDIT_FLAG	CTB: Auditing On	When set to YES, Auditing Service is enabled; all defined audit events can be audited.	CTB_YES_NO	Y
CTB_AU_BREAK_THE_ GLASS	CTB: Audit Break the Glass	Audit events of this type represent attempts to break-the-glass.	CTB_YES_NO	Y
CTB_AU_DELETE_ RESPONSIBILITY	CTB: Audit Delete Responsibility	Audit events of this type represent attempts to delete security responsibilities.	CTB_YES_NO	Y
CTB_AU_DELETE_RULE	CTB: Audit Delete Rule	Audit events of this type represent attempts to delete security rules.	CTB_YES_NO	Y
CTB_AU_DELETE_USER	CTB: Audit Delete User	Audit events of this type represent attempts to delete system or user accounts.	CTB_YES_NO	Y
CTB_AU_GENERATE_MSG	CTB: Audit Generate Message	Audit events of this type represent attempt to generate a message.	CTB_YES_NO	Y
CTB_AU_PHI_QUERY	CTB: Audit PHI Query	Audit events of this type represent querying of personal health information.	CTB_YES_NO	Y
CTB_AU_PHI_UPDATE	CTB: Audit PHI Update	Audit events of this type represent update of personal health information.	CTB_YES_NO	Y
CTB_AU_READ_ RESPONSIBILITY	CTB: Audit Read Responsibility	Audit events of this type represent attempts to view security responsibilities.	CTB_YES_NO	Y
CTB_AU_READ_RULE	CTB: Audit Read Rule	Audit events of this type represent attempts to view security rules.	CTB_YES_NO	Y

Table D–1 (Cont.) Profile Options

Code	Name	Description	Value Type Code	Default Value[1]
CTB_AU_READ_USER	CTB: Audit Read User	Audit events of this type represent attempts to view system or user accounts.	CTB_YES_NO	Y
CTB_AU_RECEIVE_MSG	CTB: Audit Receive Message	Audit events of this type represent attempt to receive a message.	CTB_YES_NO	Y
CTB_AU_SEND_ACK	CTB: Audit Send Acknowledgement	Audit events of this type represent attempt to send an acknowledgement.	CTB_YES_NO	Y
CTB_AU_SEND_MSG	CTB: Audit Send Message	Audit events of this type represent attempt to send a message.	CTB_YES_NO	Y
CTB_AU_SESSION_END	CTB: Audit Session End	Audit events of this type represent end of session.	CTB_YES_NO	Y
CTB_AU_SESSION_START	CTB: Audit Session Start	Audit events of this type represent start of sessions.	CTB_YES_NO	Y
CTB_AU_SKIP_MSG	CTB: Audit Skip Message	Audit events of this type represent attempt to skip a message.	CTB_YES_NO	Y
CTB_AU_SUBMIT_TRANS	CTB: Audit Submit Transaction	Audit events of this type represent attempt to submit a transaction.	CTB_YES_NO	Y
CTB_AU_UPDATE_ RESPONSIBILITY	CTB: Audit Update Responsibility	Audit events of this type represent attempts to update security responsibilities.	CTB_YES_NO	Y
CTB_AU_UPDATE_RULE	CTB: Audit Update Rule	Audit events of this type represent attempts to update security rules.	CTB_YES_NO	Y
CTB_AU_UPDATE_USER	CTB: Audit Update User	Audit events of this type represent attempts to update system or user accounts.	CTB_YES_NO	Y
CTB_LOGIN_URL	CTB: Login URL	URL to log in to HTB.	TEXT	N
CTB_OMP_HTB_OID	HTB OID	Identifies the sending organization as part of the message wrapper in outbound messages. Default value is set to blank or nULL, and must be set during the implementation of Outbound Messaging Services.	TEXT	N
CTB_OMP_HTB_ORG_ID	Htb Organization ID	Identifies the internal HTB organization identifier. Default value is set to blank or NULL and must be set during the implementation of Outbound Messaging Services—upon creation of the HTB organization.	TEXT	N

Table D–1 (Cont.) Profile Options

Code	Name	Description	Value Type Code	Default Value[1]
CTB_OS_BUSINESS_GROUP_ID	CTB: Business Group ID	Sets the default business group for the creation of organizations and internal addresses.	HR_ORGANIZATIONS_ALL	N
CTB_PI_ENABLED_DOMAIN	CTB: Enabled Domain	Enables or disables Domain Services.	TEXT	N
DQM_IDENTIFICATION_NAME	DQM Identification Name	Identifies which Identification Type/Issuing Authority combination is passed to DQM for matching.	CTB_PI_PARTY_ID_TYPES	N
DUPLICATE_THRESHOLD	Duplicate Threshold	Identifies Duplicate Threshold for matching.	NUMBER	N
EXTERNAL_PERSON_ID_ATTRIBUTE	External Person ID Custom Attribute	Identifies which ATTRIBUTEXXX is used for external person id in HZ_PARTIES (used in batch correlation for temporary storage of external IDs)	TEXT	N
EXTERNAL_SYSTEM_ID_ATTRIBUTE	External System ID Custom Attribute	Identifies which ATTRIBUTEXXX is used for external person system in HZ_PARTIES (used in batch correlation for temporary storage of external IDs).	TEXT	N
IDENTIFICATION_ATTRIBUTE	Identification Custom Attribute	Identifies which CUSTOM_ATTRIBUTEXXX is used for Identification Number.	TEXT	N
PHONE_STATUS_ATTRIBUTE	Phone Status Custom Attribute	Identifies which CUSTOM_ATTRIBUTEXXX is used for HZ_CONTACT_POINTS.STATUS.	TEXT	N
PHONE_USAGE_TYPE_ATTRIBUTE	Phone Usage Type Custom Attribute	Identifies which CUSTOM_ATTRIBUTEXXX is used for HZ_CONTACT_POINTS.CONTACT_POINT_PURPOSE.	TEXT	N
RULE_NAME	Rule Name	Identifies which Rules set to use for auto-matching within Create and Correlate interfaces.	TEXT	N
SEARCH_RULE_NAME	CTB: Search Rule Name	Identifies the rule for searching.	TEXT	N

[1] Values not seeded; must be set by System Administrator during initial implementation.

E

Seeded Audit Events

Table E–1 lists predefined (seeded) audit events included with the HTB platform, sorted by Audit Event Type:

Table E–1 Seeded Audit Events

Event Type	Description
Security Services	...
CTB_AU_ADD_RESPONSIBILITY	Add Responsibility Event
CTB_AU_ADD_RULE	Add Rule Event
CTB_AU_ADD_USER	Add User Event
CTB_AU_BREAK_THE_GLASS	Set Context Attribute Event
CTB_AU_DELETE_RESPONSIBILITY	Delete Responsibility Event
CTB_AU_DELETE_RULE	Delete Rule Event
CTB_AU_DELETE_USER	Delete User Event
CTB_AU_PHI_QUERY	Personal Health Information Query Event
CTB_AU_PHI_UPDATE	Personal Health Information Update Event
CTB_AU_READ_RESPONSIBILITY	Read Responsibility Event
CTB_AU_READ_RULE	Read Rule Event
CTB_AU_READ_USER	Read User Event
CTB_AU_SESSION_END	Session End Event
CTB_AU_SESSION_START	Session Start Event
CTB_AU_UPDATE_RESPONSIBILITY	Update Responsibility Event
CTB_AU_UPDATE_RULE	Update Rule Event
CTB_AU_UPDATE_USER	Update User Event
Messaging Services	...
CTB_AU_RECEIVE_MSG	Receive Message Event
CTB_AU_SEND_ACK	Send Acknowledgement Event
CTB_AU_SEND_MSG	Send Message Event
CTB_AU_SKIP_MSG	Skip Message Event
CTB_AU_SUBMIT_TRANS	Submit Transaction Event

F

Seeded ID Types

Table F–1 lists ID Types that are seeded in the `CTB_PI_PARTY_ID_TYPES` table, included with the HTB Platform:

Table F–1 Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	Afghanistan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Albania Birth Certificate
BIRTH CERTFCT	Birth Certificate	Algeria Birth Certificate
BIRTH CERTFCT	Birth Certificate	American Samoa Birth Certificate
BIRTH CERTFCT	Birth Certificate	Andorra Birth Certificate
BIRTH CERTFCT	Birth Certificate	Angola Birth Certificate
BIRTH CERTFCT	Birth Certificate	Anguilla Birth Certificate
BIRTH CERTFCT	Birth Certificate	Antarctica Birth Certificate
BIRTH CERTFCT	Birth Certificate	Antigua and Barbuda Birth Certificate
BIRTH CERTFCT	Birth Certificate	Argentina Birth Certificate
BIRTH CERTFCT	Birth Certificate	Armenia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Aruba Birth Certificate
BIRTH CERTFCT	Birth Certificate	Australia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Austria Birth Certificate
BIRTH CERTFCT	Birth Certificate	Azerbaijan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bahamas Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bahrain Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bangladesh Birth Certificate
BIRTH CERTFCT	Birth Certificate	Barbados Birth Certificate
BIRTH CERTFCT	Birth Certificate	Belarus Birth Certificate
BIRTH CERTFCT	Birth Certificate	Belgium Birth Certificate
BIRTH CERTFCT	Birth Certificate	Belize Birth Certificate
BIRTH CERTFCT	Birth Certificate	Benin Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bermuda Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bhutan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bolivia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bosnia and Herzegovina Birth Certificate
BIRTH CERTFCT	Birth Certificate	Botswana Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bouvet Island Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	Brazil Birth Certificate
BIRTH CERTFCT	Birth Certificate	British Indian Ocean Territory Birth Certificate
BIRTH CERTFCT	Birth Certificate	Brunei Darussalam Birth Certificate
BIRTH CERTFCT	Birth Certificate	Bulgaria Birth Certificate
BIRTH CERTFCT	Birth Certificate	Burkina Faso Birth Certificate
BIRTH CERTFCT	Birth Certificate	Burundi Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cambodia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cameroon Birth Certificate
BIRTH CERTFCT	Birth Certificate	Canada Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cape Verde Birth Certificate
BIRTH CERTFCT	Birth Certificate	Catalonia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cayman Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Central African Republic Birth Certificate
BIRTH CERTFCT	Birth Certificate	Chad Birth Certificate
BIRTH CERTFCT	Birth Certificate	Chile Birth Certificate
BIRTH CERTFCT	Birth Certificate	China Birth Certificate
BIRTH CERTFCT	Birth Certificate	Christmas Island Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cocos (Keeling) Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Colombia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Comoros Birth Certificate
BIRTH CERTFCT	Birth Certificate	Congo Birth Certificate
BIRTH CERTFCT	Birth Certificate	Congo, (The Democratic Republic of the) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cook Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Costa Rica Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cote d'Ivoire Birth Certificate
BIRTH CERTFCT	Birth Certificate	Croatia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cuba Birth Certificate
BIRTH CERTFCT	Birth Certificate	Cyprus Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	Czech Republic Birth Certificate
BIRTH CERTFCT	Birth Certificate	Denmark Birth Certificate
BIRTH CERTFCT	Birth Certificate	Djibouti Birth Certificate
BIRTH CERTFCT	Birth Certificate	Dominica Birth Certificate
BIRTH CERTFCT	Birth Certificate	Dominican Republic Birth Certificate
BIRTH CERTFCT	Birth Certificate	East Timor Birth Certificate
BIRTH CERTFCT	Birth Certificate	Ecuador Birth Certificate
BIRTH CERTFCT	Birth Certificate	Egypt Birth Certificate
BIRTH CERTFCT	Birth Certificate	El Salvador Birth Certificate
BIRTH CERTFCT	Birth Certificate	Equatorial Guinea Birth Certificate
BIRTH CERTFCT	Birth Certificate	Eritrea Birth Certificate
BIRTH CERTFCT	Birth Certificate	Estonia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Ethiopia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Falkland Islands (Malvinas) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Faroe Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Fiji Birth Certificate
BIRTH CERTFCT	Birth Certificate	Finland Birth Certificate
BIRTH CERTFCT	Birth Certificate	France Birth Certificate
BIRTH CERTFCT	Birth Certificate	French Guiana Birth Certificate
BIRTH CERTFCT	Birth Certificate	French Polynesia Birth Certificate
BIRTH CERTFCT	Birth Certificate	French Southern Territories Birth Certificate
BIRTH CERTFCT	Birth Certificate	Gabon Birth Certificate
BIRTH CERTFCT	Birth Certificate	Gambia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Georgia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Germany Birth Certificate
BIRTH CERTFCT	Birth Certificate	Ghana Birth Certificate
BIRTH CERTFCT	Birth Certificate	Gibraltar Birth Certificate
BIRTH CERTFCT	Birth Certificate	Greece Birth Certificate
BIRTH CERTFCT	Birth Certificate	Greenland Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	Grenada Birth Certificate
BIRTH CERTFCT	Birth Certificate	Guadeloupe Birth Certificate
BIRTH CERTFCT	Birth Certificate	Guam Birth Certificate
BIRTH CERTFCT	Birth Certificate	Guatemala Birth Certificate
BIRTH CERTFCT	Birth Certificate	Guinea Birth Certificate
BIRTH CERTFCT	Birth Certificate	Guinea-Bissau Birth Certificate
BIRTH CERTFCT	Birth Certificate	Guyana Birth Certificate
BIRTH CERTFCT	Birth Certificate	Haiti Birth Certificate
BIRTH CERTFCT	Birth Certificate	Heard Island and McDonald Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Holy See (Vatican City State) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Honduras Birth Certificate
BIRTH CERTFCT	Birth Certificate	Hong Kong Birth Certificate
BIRTH CERTFCT	Birth Certificate	Hungary Birth Certificate
BIRTH CERTFCT	Birth Certificate	Iceland Birth Certificate
BIRTH CERTFCT	Birth Certificate	India Birth Certificate
BIRTH CERTFCT	Birth Certificate	Indonesia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Iran (Islamic Republic of) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Iraq Birth Certificate
BIRTH CERTFCT	Birth Certificate	Ireland Birth Certificate
BIRTH CERTFCT	Birth Certificate	Israel Birth Certificate
BIRTH CERTFCT	Birth Certificate	Italy Birth Certificate
BIRTH CERTFCT	Birth Certificate	Jamaica Birth Certificate
BIRTH CERTFCT	Birth Certificate	Japan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Jordan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Kazakhstan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Kenya Birth Certificate
BIRTH CERTFCT	Birth Certificate	Kiribati Birth Certificate
BIRTH CERTFCT	Birth Certificate	Korea, Democratic People's Republic of Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	Korea, Republic of Birth Certificate
BIRTH CERTFCT	Birth Certificate	Kuwait Birth Certificate
BIRTH CERTFCT	Birth Certificate	Kyrgyzstan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Lao People's Democratic Republic Birth Certificate
BIRTH CERTFCT	Birth Certificate	Latvia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Lebanon Birth Certificate
BIRTH CERTFCT	Birth Certificate	Lesotho Birth Certificate
BIRTH CERTFCT	Birth Certificate	Liberia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Libyan Arab Jamahiriya Birth Certificate
BIRTH CERTFCT	Birth Certificate	Liechtenstein Birth Certificate
BIRTH CERTFCT	Birth Certificate	Lithuania Birth Certificate
BIRTH CERTFCT	Birth Certificate	Luxembourg Birth Certificate
BIRTH CERTFCT	Birth Certificate	Macau Birth Certificate
BIRTH CERTFCT	Birth Certificate	Macedonia, (The Former Yugoslav Republic of) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Madagascar Birth Certificate
BIRTH CERTFCT	Birth Certificate	Malawi Birth Certificate
BIRTH CERTFCT	Birth Certificate	Malaysia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Maldives Birth Certificate
BIRTH CERTFCT	Birth Certificate	Mali Birth Certificate
BIRTH CERTFCT	Birth Certificate	Malta Birth Certificate
BIRTH CERTFCT	Birth Certificate	Marshall Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Martinique Birth Certificate
BIRTH CERTFCT	Birth Certificate	Mauritania Birth Certificate
BIRTH CERTFCT	Birth Certificate	Mauritius Birth Certificate
BIRTH CERTFCT	Birth Certificate	Mayotte Birth Certificate
BIRTH CERTFCT	Birth Certificate	Mexico Birth Certificate
BIRTH CERTFCT	Birth Certificate	Micronesia (Federated States of) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Moldova, (Republic of) Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	Monaco Birth Certificate
BIRTH CERTFCT	Birth Certificate	Mongolia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Montserrat Birth Certificate
BIRTH CERTFCT	Birth Certificate	Morocco Birth Certificate
BIRTH CERTFCT	Birth Certificate	Mozambique Birth Certificate
BIRTH CERTFCT	Birth Certificate	Myanmar Birth Certificate
BIRTH CERTFCT	Birth Certificate	Namibia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Nauru Birth Certificate
BIRTH CERTFCT	Birth Certificate	Nepal Birth Certificate
BIRTH CERTFCT	Birth Certificate	Netherlands Antilles Birth Certificate
BIRTH CERTFCT	Birth Certificate	Netherlands Birth Certificate
BIRTH CERTFCT	Birth Certificate	New Caledonia Birth Certificate
BIRTH CERTFCT	Birth Certificate	New Zealand Birth Certificate
BIRTH CERTFCT	Birth Certificate	Nicaragua Birth Certificate
BIRTH CERTFCT	Birth Certificate	Niger Birth Certificate
BIRTH CERTFCT	Birth Certificate	Nigeria Birth Certificate
BIRTH CERTFCT	Birth Certificate	Niue Birth Certificate
BIRTH CERTFCT	Birth Certificate	Norfolk Island Birth Certificate
BIRTH CERTFCT	Birth Certificate	Northern Mariana Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Norway Birth Certificate
BIRTH CERTFCT	Birth Certificate	Obsolete see CD territory Birth Certificate
BIRTH CERTFCT	Birth Certificate	Obsolete see FR territory Birth Certificate
BIRTH CERTFCT	Birth Certificate	Obsolete see LT territory Birth Certificate
BIRTH CERTFCT	Birth Certificate	Oman Birth Certificate
BIRTH CERTFCT	Birth Certificate	Pakistan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Palau Birth Certificate
BIRTH CERTFCT	Birth Certificate	Palestinian Territory, (Occupied) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Panama Birth Certificate
BIRTH CERTFCT	Birth Certificate	Papua New Guinea Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	Paraguay Birth Certificate
BIRTH CERTFCT	Birth Certificate	Peru Birth Certificate
BIRTH CERTFCT	Birth Certificate	Philippines Birth Certificate
BIRTH CERTFCT	Birth Certificate	Pitcairn Birth Certificate
BIRTH CERTFCT	Birth Certificate	Poland Birth Certificate
BIRTH CERTFCT	Birth Certificate	Portugal Birth Certificate
BIRTH CERTFCT	Birth Certificate	Puerto Rico Birth Certificate
BIRTH CERTFCT	Birth Certificate	Qatar Birth Certificate
BIRTH CERTFCT	Birth Certificate	Reunion Birth Certificate
BIRTH CERTFCT	Birth Certificate	Romania Birth Certificate
BIRTH CERTFCT	Birth Certificate	Russian Federation Birth Certificate
BIRTH CERTFCT	Birth Certificate	Rwanda Birth Certificate
BIRTH CERTFCT	Birth Certificate	Saint Helena Birth Certificate
BIRTH CERTFCT	Birth Certificate	Saint Kitts and Nevis Birth Certificate
BIRTH CERTFCT	Birth Certificate	Saint Lucia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Saint Pierre and Miquelon Birth Certificate
BIRTH CERTFCT	Birth Certificate	Saint Vincent and the Grenadines Birth Certificate
BIRTH CERTFCT	Birth Certificate	Samoa Birth Certificate
BIRTH CERTFCT	Birth Certificate	San Marino Birth Certificate
BIRTH CERTFCT	Birth Certificate	Sao Tome and Principe Birth Certificate
BIRTH CERTFCT	Birth Certificate	Saudi Arabia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Senegal Birth Certificate
BIRTH CERTFCT	Birth Certificate	Seychelles Birth Certificate
BIRTH CERTFCT	Birth Certificate	Sierra Leone Birth Certificate
BIRTH CERTFCT	Birth Certificate	Singapore Birth Certificate
BIRTH CERTFCT	Birth Certificate	Slovakia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Slovenia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Solomon Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Somalia Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	South Africa Birth Certificate
BIRTH CERTFCT	Birth Certificate	South Georgia and the South Sandwich Island Birth Certificate
BIRTH CERTFCT	Birth Certificate	Spain Birth Certificate
BIRTH CERTFCT	Birth Certificate	Sri Lanka Birth Certificate
BIRTH CERTFCT	Birth Certificate	Sudan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Suriname Birth Certificate
BIRTH CERTFCT	Birth Certificate	Svalbard and Jan Mayen Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Swaziland Birth Certificate
BIRTH CERTFCT	Birth Certificate	Sweden Birth Certificate
BIRTH CERTFCT	Birth Certificate	Switzerland Birth Certificate
BIRTH CERTFCT	Birth Certificate	Syrian Arab Republic Birth Certificate
BIRTH CERTFCT	Birth Certificate	Taiwan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Tajikistan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Tanzania, (United Republic of) Birth Certificate
BIRTH CERTFCT	Birth Certificate	Thailand Birth Certificate
BIRTH CERTFCT	Birth Certificate	Togo Birth Certificate
BIRTH CERTFCT	Birth Certificate	Tokelau Birth Certificate
BIRTH CERTFCT	Birth Certificate	Tonga Birth Certificate
BIRTH CERTFCT	Birth Certificate	Trinidad and Tobago Birth Certificate
BIRTH CERTFCT	Birth Certificate	Tunisia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Turkey Birth Certificate
BIRTH CERTFCT	Birth Certificate	Turkmenistan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Turks and Caicos Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Tuvalu Birth Certificate
BIRTH CERTFCT	Birth Certificate	Uganda Birth Certificate
BIRTH CERTFCT	Birth Certificate	Ukraine Birth Certificate
BIRTH CERTFCT	Birth Certificate	United Arab Emirates Birth Certificate
BIRTH CERTFCT	Birth Certificate	United Kingdom Birth Certificate

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
BIRTH CERTFCT	Birth Certificate	United States Birth Certificate
BIRTH CERTFCT	Birth Certificate	United States Minor Outlying Islands Birth Certificate
BIRTH CERTFCT	Birth Certificate	Uruguay Birth Certificate
BIRTH CERTFCT	Birth Certificate	Uzbekistan Birth Certificate
BIRTH CERTFCT	Birth Certificate	Vanuatu Birth Certificate
BIRTH CERTFCT	Birth Certificate	Venezuela Birth Certificate
BIRTH CERTFCT	Birth Certificate	Viet Nam Birth Certificate
BIRTH CERTFCT	Birth Certificate	Virgin Islands, British Birth Certificate
BIRTH CERTFCT	Birth Certificate	Virgin Islands, U.S. Birth Certificate
BIRTH CERTFCT	Birth Certificate	Wallis and Futuna Birth Certificate
BIRTH CERTFCT	Birth Certificate	Western Sahara Birth Certificate
BIRTH CERTFCT	Birth Certificate	Yemen Birth Certificate
BIRTH CERTFCT	Birth Certificate	Yugoslavia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Zambia Birth Certificate
BIRTH CERTFCT	Birth Certificate	Zimbabwe Birth Certificate
CANADA DRIVER LCNS	Canada Driver's License	Alberta Driver's License
CANADA DRIVER LCNS	Canada Driver's License	British Columbia Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Manitoba Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Nanavut Driver's License
CANADA DRIVER LCNS	Canada Driver's License	New Brunswick Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Newfoundland Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Northwest Territories Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Nova Scotia Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Ontario Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Prince Edward Island Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Quebec Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Saskatchewan Driver's License
CANADA DRIVER LCNS	Canada Driver's License	Yukon Territories Driver's License

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
CITIZENSHIP CARD	US Citizenship Card	US Citizenship Card
MILITARY IDNTFCTN CARD	U.S. Military Identification Card	U.S. Military Identification Card
MILITARY SEPARATION CARD	U.S. Military Separation Document	U.S. Military Separation Document
NATURALIZATION CERTFCT	U.S. Certificate of Naturalization	U.S. Certificate of Naturalization
PASSPORT	Passport	Afghanistan Passport
PASSPORT	Passport	Albania Passport
PASSPORT	Passport	Algeria Passport
PASSPORT	Passport	American Samoa Passport
PASSPORT	Passport	Andorra Passport
PASSPORT	Passport	Angola Passport
PASSPORT	Passport	Anguilla Passport
PASSPORT	Passport	Antarctica Passport
PASSPORT	Passport	Antigua and Barbuda Passport
PASSPORT	Passport	Argentina Passport
PASSPORT	Passport	Armenia Passport
PASSPORT	Passport	Aruba Passport
PASSPORT	Passport	Australia Passport
PASSPORT	Passport	Austria Passport
PASSPORT	Passport	Azerbaijan Passport
PASSPORT	Passport	Bahamas Passport
PASSPORT	Passport	Bahrain Passport
PASSPORT	Passport	Bangladesh Passport
PASSPORT	Passport	Barbados Passport
PASSPORT	Passport	Belarus Passport
PASSPORT	Passport	Belgium Passport
PASSPORT	Passport	Belize Passport
PASSPORT	Passport	Benin Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Bermuda Passport
PASSPORT	Passport	Bhutan Passport
PASSPORT	Passport	Bolivia Passport
PASSPORT	Passport	Bosnia and Herzegovina Passport
PASSPORT	Passport	Botswana Passport
PASSPORT	Passport	Bouvet Island Passport
PASSPORT	Passport	Brazil Passport
PASSPORT	Passport	British Indian Ocean Territory Passport
PASSPORT	Passport	Brunei Darussalam Passport
PASSPORT	Passport	Bulgaria Passport
PASSPORT	Passport	Burkina Faso Passport
PASSPORT	Passport	Burundi Passport
PASSPORT	Passport	Cambodia Passport
PASSPORT	Passport	Cameroon Passport
PASSPORT	Passport	Canada Passport
PASSPORT	Passport	Cape Verde Passport
PASSPORT	Passport	Catalonia Passport
PASSPORT	Passport	Cayman Islands Passport
PASSPORT	Passport	Central African Republic Passport
PASSPORT	Passport	Chad Passport
PASSPORT	Passport	Chile Passport
PASSPORT	Passport	China Passport
PASSPORT	Passport	Christmas Island Passport
PASSPORT	Passport	Cocos (Keeling) Islands Passport
PASSPORT	Passport	Colombia Passport
PASSPORT	Passport	Comoros Passport
PASSPORT	Passport	Congo Passport
PASSPORT	Passport	Congo, (The Democratic Republic of the) Passport
PASSPORT	Passport	Cook Islands Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Costa Rica Passport
PASSPORT	Passport	Cote d'Ivoire Passport
PASSPORT	Passport	Croatia Passport
PASSPORT	Passport	Cuba Passport
PASSPORT	Passport	Cyprus Passport
PASSPORT	Passport	Czech Republic Passport
PASSPORT	Passport	Denmark Passport
PASSPORT	Passport	Djibouti Passport
PASSPORT	Passport	Dominica Passport
PASSPORT	Passport	Dominican Republic Passport
PASSPORT	Passport	East Timor Passport
PASSPORT	Passport	Ecuador Passport
PASSPORT	Passport	Egypt Passport
PASSPORT	Passport	El Salvador Passport
PASSPORT	Passport	Equatorial Guinea Passport
PASSPORT	Passport	Eritrea Passport
PASSPORT	Passport	Estonia Passport
PASSPORT	Passport	Ethiopia Passport
PASSPORT	Passport	Falkland Islands (Malvinas) Passport
PASSPORT	Passport	Faroe Islands Passport
PASSPORT	Passport	Fiji Passport
PASSPORT	Passport	Finland Passport
PASSPORT	Passport	France Passport
PASSPORT	Passport	French Guiana Passport
PASSPORT	Passport	French Polynesia Passport
PASSPORT	Passport	French Southern Territories Passport
PASSPORT	Passport	Gabon Passport
PASSPORT	Passport	Gambia Passport
PASSPORT	Passport	Georgia Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Germany Passport
PASSPORT	Passport	Ghana Passport
PASSPORT	Passport	Gibraltar Passport
PASSPORT	Passport	Greece Passport
PASSPORT	Passport	Greenland Passport
PASSPORT	Passport	Grenada Passport
PASSPORT	Passport	Guadeloupe Passport
PASSPORT	Passport	Guam Passport
PASSPORT	Passport	Guatemala Passport
PASSPORT	Passport	Guinea Passport
PASSPORT	Passport	Guinea-Bissau Passport
PASSPORT	Passport	Guyana Passport
PASSPORT	Passport	Haiti Passport
PASSPORT	Passport	Heard Island and McDonald Islands Passport
PASSPORT	Passport	Holy See (Vatican City State) Passport
PASSPORT	Passport	Honduras Passport
PASSPORT	Passport	Hong Kong Passport
PASSPORT	Passport	Hungary Passport
PASSPORT	Passport	Iceland Passport
PASSPORT	Passport	India Passport
PASSPORT	Passport	Indonesia Passport
PASSPORT	Passport	Iran (Islamic Republic of) Passport
PASSPORT	Passport	Iraq Passport
PASSPORT	Passport	Ireland Passport
PASSPORT	Passport	Israel Passport
PASSPORT	Passport	Italy Passport
PASSPORT	Passport	Jamaica Passport
PASSPORT	Passport	Japan Passport
PASSPORT	Passport	Jordan Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Kazakhstan Passport
PASSPORT	Passport	Kenya Passport
PASSPORT	Passport	Kiribati Passport
PASSPORT	Passport	Korea, Democratic People's Republic of Passport
PASSPORT	Passport	Korea, Republic of Passport
PASSPORT	Passport	Kuwait Passport
PASSPORT	Passport	Kyrgyzstan Passport
PASSPORT	Passport	Lao People's Democratic Republic Passport
PASSPORT	Passport	Latvia Passport
PASSPORT	Passport	Lebanon Passport
PASSPORT	Passport	Lesotho Passport
PASSPORT	Passport	Liberia Passport
PASSPORT	Passport	Libyan Arab Jamahiriya Passport
PASSPORT	Passport	Liechtenstein Passport
PASSPORT	Passport	Lithuania Passport
PASSPORT	Passport	Luxembourg Passport
PASSPORT	Passport	Macau Passport
PASSPORT	Passport	Macedonia, (The Former Yugoslav Republic of) Passport
PASSPORT	Passport	Madagascar Passport
PASSPORT	Passport	Malawi Passport
PASSPORT	Passport	Malaysia Passport
PASSPORT	Passport	Maldives Passport
PASSPORT	Passport	Mali Passport
PASSPORT	Passport	Malta Passport
PASSPORT	Passport	Marshall Islands Passport
PASSPORT	Passport	Martinique Passport
PASSPORT	Passport	Mauritania Passport
PASSPORT	Passport	Mauritius Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Mayotte Passport
PASSPORT	Passport	Mexico Passport
PASSPORT	Passport	Micronesia (Federated States of) Passport
PASSPORT	Passport	Moldova, (Republic of) Passport
PASSPORT	Passport	Monaco Passport
PASSPORT	Passport	Mongolia Passport
PASSPORT	Passport	Montserrat Passport
PASSPORT	Passport	Morocco Passport
PASSPORT	Passport	Mozambique Passport
PASSPORT	Passport	Myanmar Passport
PASSPORT	Passport	Namibia Passport
PASSPORT	Passport	Nauru Passport
PASSPORT	Passport	Nepal Passport
PASSPORT	Passport	Netherlands Antilles Passport
PASSPORT	Passport	Netherlands Passport
PASSPORT	Passport	New Caledonia Passport
PASSPORT	Passport	New Zealand Passport
PASSPORT	Passport	Nicaragua Passport
PASSPORT	Passport	Niger Passport
PASSPORT	Passport	Nigeria Passport
PASSPORT	Passport	Niue Passport
PASSPORT	Passport	Norfolk Island Passport
PASSPORT	Passport	Northern Mariana Islands Passport
PASSPORT	Passport	Norway Passport
PASSPORT	Passport	Obsolete see CD territory Passport
PASSPORT	Passport	Obsolete see FR territory Passport
PASSPORT	Passport	Obsolete see LT territory Passport
PASSPORT	Passport	Oman Passport
PASSPORT	Passport	Pakistan Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Palau Passport
PASSPORT	Passport	Palestinian Territory, (Occupied) Passport
PASSPORT	Passport	Panama Passport
PASSPORT	Passport	Papua New Guinea Passport
PASSPORT	Passport	Paraguay Passport
PASSPORT	Passport	Peru Passport
PASSPORT	Passport	Philippines Passport
PASSPORT	Passport	Pitcairn Passport
PASSPORT	Passport	Poland Passport
PASSPORT	Passport	Portugal Passport
PASSPORT	Passport	Puerto Rico Passport
PASSPORT	Passport	Qatar Passport
PASSPORT	Passport	Reunion Passport
PASSPORT	Passport	Romania Passport
PASSPORT	Passport	Russian Federation Passport
PASSPORT	Passport	Rwanda Passport
PASSPORT	Passport	Saint Helena Passport
PASSPORT	Passport	Saint Kitts and Nevis Passport
PASSPORT	Passport	Saint Lucia Passport
PASSPORT	Passport	Saint Pierre and Miquelon Passport
PASSPORT	Passport	Saint Vincent and the Grenadines Passport
PASSPORT	Passport	Samoa Passport
PASSPORT	Passport	San Marino Passport
PASSPORT	Passport	Sao Tome and Principe Passport
PASSPORT	Passport	Saudi Arabia Passport
PASSPORT	Passport	Senegal Passport
PASSPORT	Passport	Seychelles Passport
PASSPORT	Passport	Sierra Leone Passport
PASSPORT	Passport	Singapore Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Slovakia Passport
PASSPORT	Passport	Slovenia Passport
PASSPORT	Passport	Solomon Islands Passport
PASSPORT	Passport	Somalia Passport
PASSPORT	Passport	South Africa Passport
PASSPORT	Passport	South Georgia and the South Sandwich Island Passport
PASSPORT	Passport	Spain Passport
PASSPORT	Passport	Sri Lanka Passport
PASSPORT	Passport	Sudan Passport
PASSPORT	Passport	Suriname Passport
PASSPORT	Passport	Svalbard and Jan Mayen Islands Passport
PASSPORT	Passport	Swaziland Passport
PASSPORT	Passport	Sweden Passport
PASSPORT	Passport	Switzerland Passport
PASSPORT	Passport	Syrian Arab Republic Passport
PASSPORT	Passport	Taiwan Passport
PASSPORT	Passport	Tajikistan Passport
PASSPORT	Passport	Tanzania, (United Republic of) Passport
PASSPORT	Passport	Thailand Passport
PASSPORT	Passport	Togo Passport
PASSPORT	Passport	Tokelau Passport
PASSPORT	Passport	Tonga Passport
PASSPORT	Passport	Trinidad and Tobago Passport
PASSPORT	Passport	Tunisia Passport
PASSPORT	Passport	Turkey Passport
PASSPORT	Passport	Turkmenistan Passport
PASSPORT	Passport	Turks and Caicos Islands Passport
PASSPORT	Passport	Tuvalu Passport

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
PASSPORT	Passport	Uganda Passport
PASSPORT	Passport	Ukraine Passport
PASSPORT	Passport	United Arab Emirates Passport
PASSPORT	Passport	United Kingdom Passport
PASSPORT	Passport	United States Minor Outlying Islands Passport
PASSPORT	Passport	United States Passport
PASSPORT	Passport	Uruguay Passport
PASSPORT	Passport	Uzbekistan Passport
PASSPORT	Passport	Vanuatu Passport
PASSPORT	Passport	Venezuela Passport
PASSPORT	Passport	Viet Nam Passport
PASSPORT	Passport	Virgin Islands, British Passport
PASSPORT	Passport	Virgin Islands, U.S. Passport
PASSPORT	Passport	Wallis and Futuna Passport
PASSPORT	Passport	Western Sahara Passport
PASSPORT	Passport	Yemen Passport
PASSPORT	Passport	Yugoslavia Passport
PASSPORT	Passport	Zambia Passport
PASSPORT	Passport	Zimbabwe Passport
RESIDENT ALIEN CARD	U.S. Resident Alien Card	U.S. Resident Alien Card
SOCIAL SECURITY NUM	U.S. Social Security Card	U.S. Social Security Card
TEMP RESIDENT ID CARD	U.S. Temporary Resident Identification Card	U.S. Temporary Resident Identification Card
US DRIVER LCNS	US Driver's License	Alabama Driver's License
US DRIVER LCNS	US Driver's License	Alaska Driver's License
US DRIVER LCNS	US Driver's License	Arizona Driver's License
US DRIVER LCNS	US Driver's License	Arkansas Driver's License
US DRIVER LCNS	US Driver's License	California Driver's License
US DRIVER LCNS	US Driver's License	Colorado Driver's License

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
US DRIVER LCNS	US Driver's License	Connecticut Driver's License
US DRIVER LCNS	US Driver's License	Delaware Driver's License
US DRIVER LCNS	US Driver's License	Florida Driver's License
US DRIVER LCNS	US Driver's License	Georgia Driver's License
US DRIVER LCNS	US Driver's License	Hawaii Driver's License
US DRIVER LCNS	US Driver's License	Idaho Driver's License
US DRIVER LCNS	US Driver's License	Illinois Driver's License
US DRIVER LCNS	US Driver's License	Indiana Driver's License
US DRIVER LCNS	US Driver's License	Iowa Driver's License
US DRIVER LCNS	US Driver's License	Kansas Driver's License
US DRIVER LCNS	US Driver's License	Kentucky Driver's License
US DRIVER LCNS	US Driver's License	Louisiana Driver's License
US DRIVER LCNS	US Driver's License	Maine Driver's License
US DRIVER LCNS	US Driver's License	Maryland Driver's License
US DRIVER LCNS	US Driver's License	Massachusetts Driver's License
US DRIVER LCNS	US Driver's License	Michigan Driver's License
US DRIVER LCNS	US Driver's License	Minnesota Driver's License
US DRIVER LCNS	US Driver's License	Mississippi Driver's License
US DRIVER LCNS	US Driver's License	Missouri Driver's License
US DRIVER LCNS	US Driver's License	Montana Driver's License
US DRIVER LCNS	US Driver's License	Nebraska Driver's License
US DRIVER LCNS	US Driver's License	Nevada Driver's License
US DRIVER LCNS	US Driver's License	New Hampshire Driver's License
US DRIVER LCNS	US Driver's License	New Jersey Driver's License
US DRIVER LCNS	US Driver's License	New Mexico Driver's License
US DRIVER LCNS	US Driver's License	New York Driver's License
US DRIVER LCNS	US Driver's License	North Carolina Driver's License
US DRIVER LCNS	US Driver's License	North Dakota Driver's License
US DRIVER LCNS	US Driver's License	Ohio Driver's License

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
US DRIVER LCNS	US Driver's License	Oklahoma Driver's License
US DRIVER LCNS	US Driver's License	Oregon Driver's License
US DRIVER LCNS	US Driver's License	Pennsylvania Driver's License
US DRIVER LCNS	US Driver's License	Rhode Island Driver's License
US DRIVER LCNS	US Driver's License	South Carolina Driver's License
US DRIVER LCNS	US Driver's License	South Dakota Driver's License
US DRIVER LCNS	US Driver's License	Tennessee Driver's License
US DRIVER LCNS	US Driver's License	Texas Driver's License
US DRIVER LCNS	US Driver's License	Utah Driver's License
US DRIVER LCNS	US Driver's License	Vermont Driver's License
US DRIVER LCNS	US Driver's License	Virginia Driver's License
US DRIVER LCNS	US Driver's License	Washington Driver's License
US DRIVER LCNS	US Driver's License	West Virginia Driver's License
US DRIVER LCNS	US Driver's License	Wisconsin Driver's License
US DRIVER LCNS	US Driver's License	Wyoming Driver's License
US IDNTFCTN CARD	Identification Card	Alabama Identification Card
US IDNTFCTN CARD	Identification Card	Alaska Identification Card
US IDNTFCTN CARD	Identification Card	Arizona Identification Card
US IDNTFCTN CARD	Identification Card	Arkansas Identification Card
US IDNTFCTN CARD	Identification Card	California Identification Card
US IDNTFCTN CARD	Identification Card	Colorado Identification Card
US IDNTFCTN CARD	Identification Card	Connecticut Identification Card
US IDNTFCTN CARD	Identification Card	Delaware Identification Card
US IDNTFCTN CARD	Identification Card	Florida Identification Card
US IDNTFCTN CARD	Identification Card	Georgia Identification Card
US IDNTFCTN CARD	Identification Card	Hawaii Identification Card
US IDNTFCTN CARD	Identification Card	Idaho Identification Card
US IDNTFCTN CARD	Identification Card	Illinois Identification Card
US IDNTFCTN CARD	Identification Card	Indiana Identification Card

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
US IDNTFCTN CARD	Identification Card	Iowa Identification Card
US IDNTFCTN CARD	Identification Card	Kansas Identification Card
US IDNTFCTN CARD	Identification Card	Kentucky Identification Card
US IDNTFCTN CARD	Identification Card	Louisiana Identification Card
US IDNTFCTN CARD	Identification Card	Maine Identification Card
US IDNTFCTN CARD	Identification Card	Maryland Identification Card
US IDNTFCTN CARD	Identification Card	Massachusetts Identification Card
US IDNTFCTN CARD	Identification Card	Michigan Identification Card
US IDNTFCTN CARD	Identification Card	Minnesota Identification Card
US IDNTFCTN CARD	Identification Card	Mississippi Identification Card
US IDNTFCTN CARD	Identification Card	Missouri Identification Card
US IDNTFCTN CARD	Identification Card	Montana Identification Card
US IDNTFCTN CARD	Identification Card	Nebraska Identification Card
US IDNTFCTN CARD	Identification Card	Nevada Identification Card
US IDNTFCTN CARD	Identification Card	New Hampshire Identification Card
US IDNTFCTN CARD	Identification Card	New Jersey Identification Card
US IDNTFCTN CARD	Identification Card	New Mexico Identification Card
US IDNTFCTN CARD	Identification Card	New York Identification Card
US IDNTFCTN CARD	Identification Card	North Carolina Identification Card
US IDNTFCTN CARD	Identification Card	North Dakota Identification Card
US IDNTFCTN CARD	Identification Card	Ohio Driver's Identification Card
US IDNTFCTN CARD	Identification Card	Oklahoma Identification Card
US IDNTFCTN CARD	Identification Card	Oregon Identification Card
US IDNTFCTN CARD	Identification Card	Pennsylvania Identification Card
US IDNTFCTN CARD	Identification Card	Rhode Island Identification Card
US IDNTFCTN CARD	Identification Card	South Carolina Identification Card
US IDNTFCTN CARD	Identification Card	South Dakota Identification Card
US IDNTFCTN CARD	Identification Card	Tennessee Identification Card
US IDNTFCTN CARD	Identification Card	Texas Identification Card

Table F–1 (Cont.) Seeded ID Types

Identification Type Code	ID Type	Identification Name/Description
US IDNTFCTN CARD	Identification Card	Utah Identification Card
US IDNTFCTN CARD	Identification Card	Vermont Identification Card
US IDNTFCTN CARD	Identification Card	Virginia Identification Card
US IDNTFCTN CARD	Identification Card	Washington Identification Card
US IDNTFCTN CARD	Identification Card	West Virginia Identification Card
US IDNTFCTN CARD	Identification Card	Wisconsin Identification Card
US IDNTFCTN CARD	Identification Card	Wyoming Identification Card

G

Seeded OMP Workflow Data

This appendix describes the use of seeded workflow data, including workflow processes, business events, and event subscriptions for HTB Outbound Messaging Services (OMP).

This appendix contains the following topics:

Introduction

OMP workflow processes control outbound clinical and administrative HL7 messages from Oracle Healthcare Transaction Base (HTB) by the Outbound Message Processor (OMP), as described in Section 4.13.

Oracle Workflow is a central component of this architecture and provides the management of the process—from receipt of a trigger event generated by HTB to generation of HL7 version 3 conformant messages, to delivery of HL7 messages to an external Interface Engine.

Overview of Outbound Message Processor

Figures G–1 and G–2 illustrate the OMP Workflow architecture:

Figure G–1 Overview of Trigger Event OMP Process

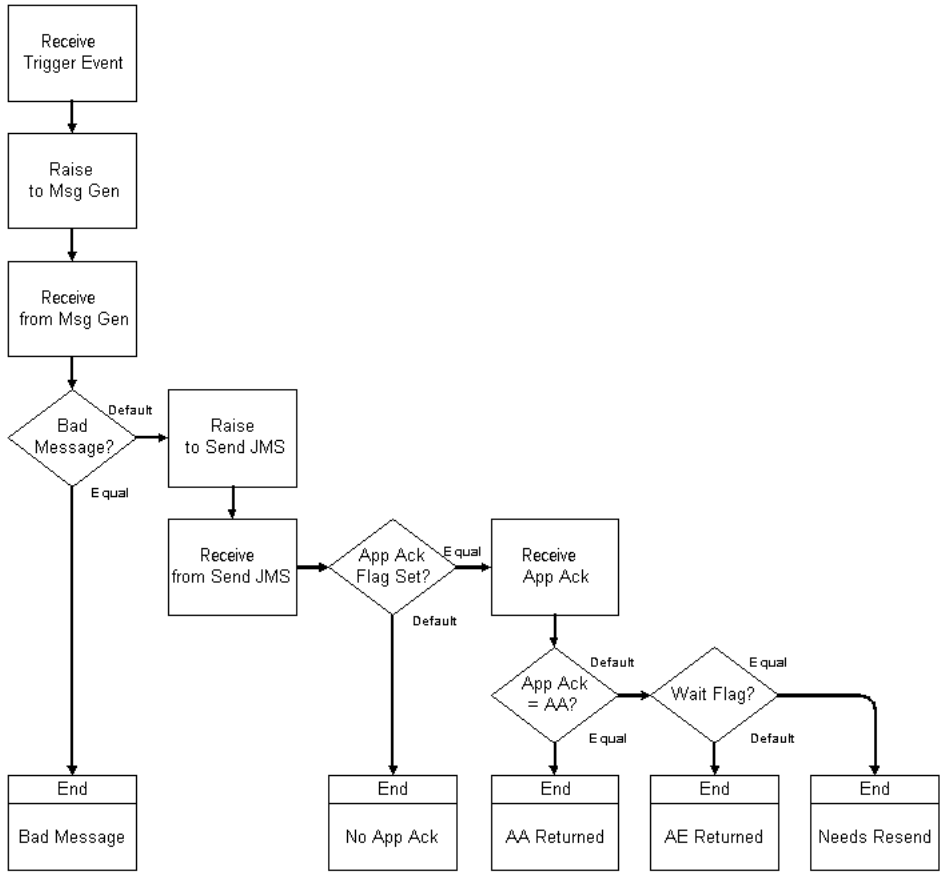
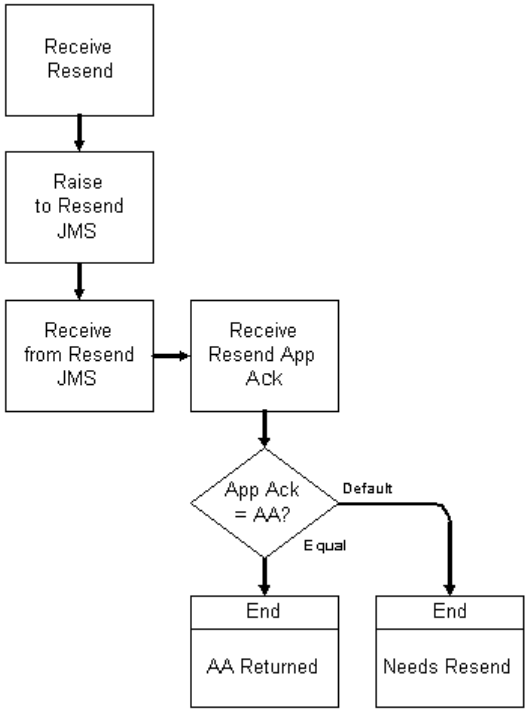


Figure G–2 Overview of Resend OMP Process



Assumptions:

- The two workflow process illustrated by Figures G–1 and G–2 need not be correlated on the basis of the Message ID.
- Each receiving organization configured as part of the OMP will have one queue configured for sending messages from the workflow.
- HTB application Administrators will monitor the error log created by OMP to review errors, and no notifications will be sent as part of the pre-configured workflow process.

Flow through the trigger event *Generate HL7 Message* process is described below:

- HTB applications raise trigger event objects as business events by calling the `generateMessage` method in the `OMPService`.

See Also: *Oracle Javadoc for HTB* `[oracle.apps.ctb.message.omprocessor, OMPService.generateMessage]`

- Upon receipt of the business event by the *Rcv Trigger Event* activity, the process transitions to the *Raise To MsgGen* event activity.
- **The** Raise To MsgGen event activity raises a business event to the message generator (`GenerateMessageSubscription`) in the Business Event System, which is implemented as a Java subscription (JBES).
- The message generator then generates the full HL7 v3 message.
- The Java subscription logs the status to Submission Unit. If there is an error, the message generator raises the exception caught by the java subscription
- The message generator raises a business event with the full HL7 message as `EVENT_MESSAGE` (if successfully generated) to the ***Rcv from MsgGen*** event activity, returning flow control back to the workflow process. It also sets `CTB_BAD_MSG` depending on the success or failure of the message generator operation (success means `CTB_BAD_MSG` = *N*; otherwise *Y*)
- If the *Check For Bad Msg* (compare text) activity finds the `CTB_BAD_MSG` attribute set to *Y*, it transitions to an End activity with a result of *Skipped*, which terminates the process. Otherwise, the process transitions to the *Raise to Send JMS* event activity.
- **The Raise to Send JMS** event activity raises a business event to the `SendJMSSubscription` in the Business Event System, which is implemented as a Java subscription (JBES).
- `SendJMSSubscription` then sends the event to a JMS AQ queue agent that enqueues the message. This queue is configured with the `wf_event_ojmstext_qh` queue handler, which converts the business event to a JMS text message.
- The `MessageDeliveryAgent` (HTB Gateway) removes it from the associated queue based on the communication channel (configured as part of the HTB Gateway) and delivers the message to its destination.
- The `MessageDeliveryAgent` then raises a business event to the *Rcv from Send JMS* event activity, returning flow control back to the workflow process, and transitions to the *App Ack Flag Set* activity.
- **The** App Ack Flag Set (compare text) activity examines the `APP_ACK_FLAG` received in **Rcv Trigger Event** activity at the beginning of the workflow process.

- If APP_ACK_FLAG is *N* (based on None value set as part of the configuration for that receiving org), the workflow instance terminates with a result of *No App Ack*.
- If APP_ACK_FLAG is *Y* (based on Application value set as part of the configuration for that receiving org), flow transitions to the ***Rcv App Ack*** event activity, which waits for a business event containing the application acknowledgement to be raised to it from the OMPService. The AcknowledgeMessage method is called by the interface engine through the HTB Gateway.
- After the application acknowledgement is received by the Rcv App Ack event activity, flow transitions to *App Ack is AA* (compare text) activity, where the ACK_TYPE_CD attribute is examined to determine whether the acknowledgement is an *AA* (success) or an *AE* (application error).
- If the ACK_TYPE_CD attribute is *AA*, the process terminates with a result of *AA Returned*.
- If the ACK_TYPE_CD attribute is *AE*, flow transitions to ***Check Wait Flag*** (compare text) activity, which examines the WAIT_FOR_ACK attribute.
- If the WAIT_FOR_ACK attribute is *N (false)*, the process transitions to an **End** activity with a result of *AE Returned*, which terminates the process.
- If the WAIT_FOR_ACK attribute is *Y (true)*, flow transitions to another **End** activity with a result of *Needs Resend*, which terminates the process.

Scheduled message deliveries are managed by using the *deferred subscription processing* feature of the event manager as described in the *Oracle Workflow Developers Guide*. Essentially, the desired future date/time is specified in the business event SEND_DATE attribute. This results in the event being added to the WF_DEFERRED queue, where it will not be available for dequeuing until the SEND_DATE has arrived and the deferred queue agent next runs.

A site administrator monitors the error log for errors. For certain types of errors, the administrator will attempt to correct the HL7 message and resend it via the Resend workflow process. Flow through the *Resend HL7 Message* process occurs as described below:

- An application raises the corrected HL7 message as a business event by calling the resendMessage method (OMPService).

> **See Also: :** *Oracle Javadoc for HTB*
> [oracle.apps.ctb.message.omprocessor, OMPService.resendMessage]

- Process flows to the *Raise to Resend JMS* event activity, similar to the Raise to Send JMS event activity.
- The Raise to Resend JMS event activity raises a business event to the `ResendJMSSubscription` in the Business Event System, which is implemented as a Java subscription (JBES).
- `ResendJMSSubscription` sends the event to a JMS AQ queue agent which enqueues the message. This queue is configured with the `wf_event_ojmstext_qh` queue handler, which converts the business event to a JMS text message.
- The `MessageDeliveryAgent` (HTB Gateway) dequeues it from the associated queue based on the communication channel (configured as part of the HTB Gateway) and delivers the message to its destination.
- The `MessageDeliveryAgent` raises a business event to the Rcv from Send JMS event activity, returning flow control back to the workflow process.
- In the Resend HL7 Message process, the `APP_ACK_FLAG` should be *Y* as received in Rcv Trigger Event activity at the beginning of the first workflow process; otherwise the Resend HL7 Message process is not initiated.
- The process flows to the *Rcv Resend App Ack* event activity, which waits for a business event containing the application acknowledgement to be raised to it from the `OMPService. AcknowledgeMessage` method called by the interface engine through the HTB Gateway.
- After the application acknowledgement is received by the Rcv Resend App Ack event activity, flow transitions to App Ack is AA (compare text) activity where the `ACK_TYPE_CD` attribute is examined to determine whether the acknowledgement is an *AA* (success) or an *AE* (application error).
- If the `ACK_TYPE_CD` attribute is *AA*, the process terminates with a result of *AA Returned.*
- If the `ACK_TYPE_CD` attribute is *AE*, flow transitions to another **End** activity with a result of *Needs Resend*, which terminates the process.

Seeded Events and Subscriptions

HL7 Trigger Event

This business event can be raised by an HTB application or Inbound Message Processing, using the `generateMessage` method to send the event message to and start the Generate HL7 Message *(GEN_MSG)* process in the CTB OMP item

type. The business event type is defined through the Oracle Workflow Web Application user interface, with the default event type name `oracle.apps.ctb.message.omprocessor.event.TriggerEvent`. The event key of the trigger event is used as the correlation ID for the workflow process instance throughout the whole process and is stored in `MSG_ID` attribute.

If the message delivery is to be scheduled for a future date/time, the `SEND_DATE` attribute must be set with the desired delivery date/time in the business event (`oracle.apps.ctb.message.omprocessor.event.TriggerEvent`). `SEND_DATE` is produced by the `OMPService` and involves some combination of current date/time, specified date/time, and time offset. Note that a priority of `STAT` also results in a `SEND_DATE` time of *immediately*.

The `TRG_EVNT, INTERACTION_ID, RCV_ORG, FOCAL_OBJ_ID, FOCAL_OBJ_ID_VER, SECONDARY_OBJ_ID` and `SECONDARY_OBJ_ID_VER` attribute values are supplied by the `OMPService`. The `OMPService` also looks up the `APP_ACK_FLG` and `WAIT_FOR_ACK`, based on `TRG_EVNT` and `RCV_ORG`. The `SUBMIT_UNIT_ID` is also determined by the `OMPService` interface when the submission unit is created.

> **See Also:** Section 4.13, Implementing Outbound Messaging Services

- **Internal Name:** `oracle.apps.ctb.message.omprocessor.event.TriggerEvent`
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the HL7 trigger event. This subscription sends the event message to the Generate HL7 Message (`GEN_MSG`) process in the CTB OMP item type when the HL7 trigger event is raised locally. This subscription is enabled by default. Table G–1 lists the properties defined for this subscription:

Table G–1 HL7 Trigger Event Subscription Properties

Subscription Property	Value
System	`<local system>`
Source Type	`Local`

Table G–1 (Cont.) HL7 Trigger Event Subscription Properties

Subscription Property	Value
Event Filter	oracle.apps.ctb.message.omprocessor.event.TriggerEvent
Phase	10
Status	Enabled
Rule Data	Key
Customization Level	Limit
Rule Function	wf_rule.default_rule
Workflow Item Type	CTBOMP
Workflow Process Name	GEN_MSG
Parameters	None
Owner Name	CTB OMP
Owner Tag	OMP

To Message Generator

OMP workflow raises this business event from the Generate HL7 Message (GEN_MSG) process in the CTB OMP item type to the Business Event System to trigger the message generation Java subscription (java://oracle.apps.ctb.message.omprocessor.client.GenerateMessageSubscription) for HL7 message generation. Within the Java Subscription, various OMP message processors are used to access HTB to pull necessary attributes for messages, check validity, and formulate outbound version 3 messages. Messages are generated based on the TRG_EVNT, INTERACTION_ID, RCV_ORG, FOCAL_OBJ_ID, SECONDARY_OBJ_ID and other attributes sent with the business event message as parameters.

The default event name is oracle.apps.ctb.messge.omprocessor.event.ToMsgGen. The event key is set to the value of the Message ID (MSG_ID) to maintain the process Correlation ID in the message generation Java subscription and Receive from Message Generation Event. Aside from generating the HL7 message, it is the responsibility of the Java subscription to log Submission Unit and any errors encountered. The Java subscription must also set the SKIP_FLAG attribute in the business event it raises based on the exceptions raised by the message generator. *Y* indicates an error was encountered and a valid message could not be generated. *N* indicates the no problems were encountered.

- **Internal Name:** `oracle.apps.ctb.messge.omprocessor.event.ToMsgGen`
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the To Message Generator Event. This subscription sends the event message to the Generate Message Java Subscription when the To Message Generator Event is raised locally. This subscription is enabled by default. Table G–2 lists the properties defined for this subscription:

Table G–2 To Message Generator Event Subscription Properties

Subscription Property	Value
System	`<local system>`
Source Type	`Local`
Event Filter	`oracle.apps.ctb.message.omprocessor.event.ToMsgGen`
Phase	`10`
Status	`Enabled`
Rule Data	`Key`
Customization Level	`Limit`
Rule Function	`java://oracle.apps.ctb.message.omprocessor.client.GenerateMessageSubscription`
Workflow Item Type	`None`
Workflow Process Name	`None`
Parameters	`None`
Owner Name	`CTB OMP`
Owner Tag	`OMP`

From Message Generator

This business event is raised from the Generate Message Java Subscription after the message generation process is completed, and returns execution to the workflow process. The generated message is placed in the Business Event Message attribute (`EVENT_MESSAGE`) of the workflow process. The result of the message generation is also indicated by the parameter list attribute, Skip Flag (`SKIP_FLAG`).

The default event name is `oracle.apps.ctb.message.omprocessor.event.FromMsgGen` and the event key is set to the value of the Message ID by the message generation Java subscription. The event key is used as Correlation ID by the workflow engine to identify the process instance, from which the To Message Generator Event is initialized.

- **Internal Name:** oracle.apps.ctb.message.omprocessor.event.FromMsgGen
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the From Message Generator Event. This subscription sends the event message to the **Generate HL7 Message (**`GEN_MSG`**)** process in the CTB OMP item type when the From Message Generator Event is raised locally. This subscription is enabled by default. Table G–3 lists the properties defined for this subscription:

Table G–3 From Message Generator Event Subscription Properties

Subscription Property	Value
System	`<local system>`
Source Type	`Local`
Event Filter	`oracle.apps.ctb.message.omprocessor.event.FromMsgGen`
Phase	`10`
Status	`Enabled`
Rule Data	`Key`
Customization Level	`Limit`
Rule Function	`wf_rule.default_rule`
Workflow Item Type	`CTBOMP`
Workflow Process Name	`GEN_MSG`
Parameters	`None`
Owner Name	`CTB OMP`
Owner Tag	`OMP`

To Send JMS

OMP workflow raises this business event from the Generate HL7 Message (`GEN_MSG`) process in the CTB OMP item type to the Business Event System to trigger the Send JMS Java subscription (`java://oracle.apps.ctb.message.omprocessor.client.SendJMSSubscription`) to send the event to a JMS AQ queue agent that enqueues the message. This queue is configured with the `wf_event_ojmstext_qh` queue handler, which converts the business event to a JMS text message.

The `MessageDeliveryAgent` (HTB Gateway) dequeues it from the associated queue based on the communication channel (configured as part of the HTB Gateway) and delivers the message to its destination.

The default event name is `oracle.apps.ctb.message.omprocessor.event.ToSendJMS`. The event key is set to the value of the Message ID (`MSG_ID`) to maintain the process Correlation ID in the Send JMS Java subscription and Rcv from Send JMS Event.

- **Internal Name:** `oracle.apps.ctb.message.omprocessor.event.ToSendJMS`
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the *To Send JMS* Event. This subscription sends the event message to the Send JMS Java Subscription when the To Send JMS Event is raised locally. This subscription is enabled by default. Table G–4 lists the properties defined for this subscription:

Table G–4 To Send JMS Event Subscription Properties

Subscription Property	Value
System	`<local system>`
Source Type	`Local`
Event Filter	`oracle.apps.ctb.message.omprocessor.event.ToSendJMS`
Phase	`10`

Table G–4 (Cont.) To Send JMS Event Subscription Properties

Subscription Property	Value
Status	Enabled
Rule Data	Key
Customization Level	Limit
Rule Function	java://oracle.apps.ctb.message.omprocessor.client.SendJMSSubscription
Workflow Item Type	None
Workflow Process Name	None
Parameters	None
Owner Name	CTB OMP
Owner Tag	OMP

From Send JMS

This business event is raised from the Send JMS Java Subscription after the message has been delivered to the destination queue and returns the execution to the workflow process.

The default event name is `oracle.apps.ctb.message.omprocessor.event.FromSendJMS` and the event key is set to the value of the Message ID by the Send JMS Java subscription. The event key is used as Correlation ID by the workflow engine to identify the process instance, from which the To Send JMS Event is initialized.

- **Internal Name:** `oracle.apps.ctb.message.omprocessor.event.FromSendJMS`
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the From Send JMS Event. This subscription sends the event message to the Generate HL7 Message **(**`GEN_MSG`**)** process in the CTB OMP item type when the From Send JMS Event is raised locally.

This subscription is enabled by default. Table G–5 lists the properties defined for this subscription:

Table G–5 From Send JMS Event Subscription Properties

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.ctb.message.omprocessor.event.FromSendJMS
Phase	10
Status	Enabled
Rule Data	Key
Customization Level	Limit
Rule Function	wf_rule.default_rule
Workflow Item Type	CTBOMP
Workflow Process Name	GEN_MSG
Parameters	None
Owner Name	CTB OMP
Owner Tag	OMP

App Ack from IE

This business event containing the application acknowledgement can be raised from the receiving organization via the HTB gateway through the interface engine and the OMPService. This event is in response to the sending of an HL7 message through the To Send JMS event activity earlier in the Generate HL7 Message (GEN_MSG) process in the CTB OMP item type. The receiving organization sends the application acknowledgement to the interface engine (or the interface engine generates its own acknowledgement under certain error conditions). The interface engine passes the acknowledgement to the HTB Gateway and through the OMPService.AcknowledgeMessage method, which raises a business event after setting the specified attributes.

The default event name is oracle.apps.ctb.message.omprocessor.event.AppAck and the event key is set to the value of the Message ID when HTB Gateway calls

OMPService.AcknowledgeMessage, which is sent to the interface engine by the To Send JMS event. The MSG_ID doubles as the HL7 message identifier and the item key (and therefore the correlation ID) that the event manager uses to identify the correct workflow process instance to raise the business event back to.

Before OMPService can raise the business event, it must extract three attributes from the Application Level Acknowledgement message: (i) ACK_TYPE_CD contains the acknowledgement code of either *AA* or *AE*; (ii) ACK_ERROR_CD contains the error code (if any): and (iii) ACK_NOTE_TXT contains additional textual error information.

- **Internal Name:** oracle.apps.ctb.message.omprocessor.event.AppAck
- **Generate Function:** None
- ***Owner Name:*** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the Send App Ack from IE Event. The Workflow Engine uses the event key with the value of the Message ID as Correlation ID to decide which workflow process instance can receive the event message sent by the subscription. The subscription sends the event message to the Generate HL7 Message **(**GEN_MSG**)** process in the CTB OMP item type when the App Ack from IE Event is raised locally. This subscription is enabled by default. Table G–6 lists the properties defined for this subscription:

Table G–6 App Ack from IE Event Subscription Properties

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.ctb.message.omprocessor.event.AppAck
Phase	10
Status	Enabled
Rule Data	Key
Customization Level	Limit
Rule Function	wf_rule.default_rule
Workflow Item Type	CTBOMP

Table G–6 (Cont.) App Ack from IE Event Subscription Properties

Subscription Property	Value
Workflow Process Name	GEN_MSG
Parameters	None
Owner Name	CTB OMP
Owner Tag	OMP

Resend HL7 Message

This business event containing the modified HL7 message can be raised from an HTB application, which corrects erroneous messages. This event is in response to receiving an AE application acknowledgement from the receiving organization. The modifying application accesses the erroneous HL7 message in the error log (through the SubmissionUnit Id) and corrects it. The application passes the modified HL7 message through the OMPService.resendMessage method, which raises a business event after setting the specified attributes.

The default event name is oracle.apps.ctb.message.omprocessor.event.Resend and the event key is set to the value of HL7 message MSG_ID.

- **Internal Name:** oracle.apps.ctb.message.omprocessor.event.Resend
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the Resend HL7 Message Event. This subscription sends the event message to the Resend HL7 Message (RESEND_MSG) process in the CTB OMP item type when the Resend HL7 Message Event is raised locally. This subscription is enabled by default. Table G–7 lists the properties defined for this subscription:

Table G–7 Resend HL7 Message Event Subscription Properties

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.ctb.message.omprocessor.event.Resend
Phase	10
Status	Enabled
Rule Data	Key
Customization Level	Limit
Rule Function	wf_rule.default_rule
Workflow Item Type	CTBOMP
Workflow Process Name	RESEND_MSG
Parameters	None
Owner Name	CTB OMP
Owner Tag	OMP

To Resend JMS

OMP workflow raises this business event from the Resend HL7 Message (RESEND_MSG) process in the CTB OMP item type to the Business Event System to trigger the Resend JMS Java subscription (java://oracle.apps.ctb.message.omprocessor.client.ResendJMSSubscription) to send the event to a JMS AQ queue agent that enqueues the message. This queue is configured with the wf_event_ojmstext_qh queue handler, which converts the business event to a JMS text message.

The MessageDeliveryAgent (HTB Gateway) dequeues it from the associated queue based on the communication channel (configured as part of the HTB Gateway) and delivers the message to its destination.

The default event name is oracle.apps.ctb.message.omprocessor.event.ToResendJMS. The event key is set to the value of the Message ID (MSG_ID) to maintain the process Correlation ID in the Resend JMS Java subscription and Receive from Resend JMS Event.

- **Internal Name:** `oracle.apps.ctb.message.omprocessor.event.ToResendJMS`
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the To Resend JMS Event. This subscription sends the event message to the Resend JMS Java Subscription when the To Resend JMS Event is raised locally. This subscription is enabled by default. Table G–8 lists the properties defined for this subscription:

Table G–8 To Resend JMS Event Subscription Properties

Subscription Property	Value
System	`<local system>`
Source Type	`Local`
Event Filter	`oracle.apps.ctb.message.omprocessor.event.ToResendJMS`
Phase	`10`
Status	`Enabled`
Rule Data	`Key`
Customization Level	`Limit`
Rule Function	`java://oracle.apps.ctb.message.omprocessor.client.ResendJMSSubscription`
Workflow Item Type	`None`
Workflow Process Name	`None`
Parameters	`None`
Owner Name	`CTB OMP`
Owner Tag	`OMP`

From Resend JMS

This business event is raised from the Resend JMS Java Subscription after the message has been delivered to the destination queue and returns execution to the workflow process.

The default event name is `oracle.apps.ctb.message.omprocessor.event.FromResendJMS` and the event key is set to the value of the Message ID by the Resend JMS Java subscription. The event key is used as Correlation ID by the workflow engine to identify the process instance, from which the To Resend JMS Event is initialized.

- **Internal Name:** `oracle.apps.ctb.message.omprocessor.event.FromResendJMS`
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the From Resend JMS Event. This subscription sends the event message to the Resend HL7 Message (`RESEND_MSG`) process in the CTB OMP item type when the From Resend JMS Event is raised locally. This subscription is enabled by default. Table G–9 lists the properties defined for this subscription:

Table G–9 From Resend JMS Event Subscription Properties

Subscription Property	Value
System	`<local system>`
Source Type	`Local`
Event Filter	`oracle.apps.ctb.message.omprocessor.event.FromResendJMS`
Phase	`10`
Status	`Enabled`
Rule Data	`Key`
Customization Level	`Limit`
Rule Function	`wf_rule.default_rule`
Workflow Item Type	`CTBOMP`
Workflow Process Name	`RESEND_MSG`
Parameters	`None`
Owner Name	`CTB OMP`
Owner Tag	`OMP`

Resend App Ack from IE

This business event containing the application acknowledgement can be raised from the receiving organization via the HTB gateway through the interface engine and the `OMPService`. This event is in response to the resending of the HL7 message through the To Resend JMS event activity earlier in the Resend HL7 Message **(**`RESEND_MSG`**)** process in the CTB OMP item type. The receiving organization sends the application acknowledgement to the interface engine (or the interface engine generates its own under certain error conditions). The interface engine passes the acknowledgement to the HTB Gateway and through the `OMPService.AcknowledgeMessage` method, which raises a business event after setting the specified attributes.

The default event name is `oracle.apps.ctb.message.omprocessor.event.ResendAppAck` and the event key is set to the value of the Message ID when the HTB Gateway calls `OMPService.AcknowledgeMessage`, which is sent to the interface engine by the To Resend JMS event. The `MSG_ID` doubles as the HL7 message identifier and as the item key (and therefore the correlation ID) that the event manager uses to identify the correct workflow process instance to raise the business event back to.

Before the `OMPService` can raise the business event, it must extract three attributes from the Application Level Acknowledgement message: (i) `ACK_TYPE_CD` contains the acknowledgement of either *AA* or *AE*; (ii) `ACK_ERROR_CD` contains the error code (if any); and (iii) `ACK_NOTE_TXT` contains additional textual error information.

- **Internal Name:** `oracle.apps.ctb.message.omprocessor.event.ResendAppAck`
- **Generate Function:** None
- **Owner Name:** CTB OMP
- **Owner Tag:** OMP
- **Customization Level:** Limit

OMP workflow provides one default subscription to the Resend App Ack from the IE Event. The Workflow Engine uses the event key with the value of the Message ID as Correlation ID to decide which workflow process instance can receive the event message sent by the subscription. The subscription sends the event message to the Resend HL7 Message **(**`RESEND_MSG`**)** process in the CTB OMP item type when the App Ack from IE Event is raised locally. This subscription is enabled by default. Table G–10 lists the properties defined for this subscription:

Table G–10 Resend APP Ack from IE Event Subscription Properties

Subscription Property	Value
System	<local system>
Source Type	Local
Event Filter	oracle.apps.ctb.message.omprocessor.event.ResendAppAck
Phase	10
Status	Enabled
Rule Data	Key
Customization Level	Limit
Rule Function	wf_rule_default_rule
Workflow Item Type	CTBOMP
Workflow Process Name	RESEND_MSG
Parameters	None
Owner Name	CTB OMP
Owner Tag	OMP

Configuring Oracle Workflow and the Business Event System

Configuring Oracle Workflow

The CTB OMP workflow process definition seed file (`ctbomp.wft`) is located at: `ctb/src/patch/115/import/US`

There are several options to upload process definitions from an input seed data flat file to a database or download the process definition of one or more item types from a database to an output file for further modification and upgrade.

Option #1 Use the File Open menu options in Oracle Workflow Builder to upload the workflow seed data file (`ctbomp.wft`) for process definition into database, or the File Save menu to download the item types into a flat file with an extension of `.wft` (download process definition from a database is not recommended). When saving the process definition into a database, you must provide a user name and password to the APPS schema and database connection.

Option #2 Run the Workflow Definitions Loader concurrent program from the command line by entering the following commands:

- **To upgrade:** `WFLOAD apps/pwd 0 Y UPGRADE file.wft`
- **To upload:** `WFLOAD apps/pwd 0 Y UPLOAD file.wft`
- **To download:** `WFLOAD apps/pwd 0 Y DOWNLOAD file.wft ITEMTYPE1 [ITEMTYPE2 ...ITEMTYPEN]`

Note #1: Replace `apps/pwd` with user name and password to the APPS schema; replace `file.wft` with the file specification of a workflow process definition file (`ctbomp.wft` for the CTB OMP workflow process); and replace `ITEMTYPE1, ITEMTYPE2, ...ITEMTYPEN` with the item types you want to download. You can also download all item types simultaneously by replacing `ITEMTYPE1` with `'*'` (make sure you enclose the asterisk in single quotes).

Note #2: A file specification is defined as: `@<application_short_name>:[<dir>/.../]file.ext` or `<native path>`

See Also: *Oracle Workflow Administrator's Guide, Using the Workflow Definitional Loader*

Option #3 Run the Workflow Definitions Loader for the version of Oracle Workflow embedded in Oracle Applications.

1. Navigate to the Submit Requests form in Oracle Applications to submit the Workflow Definitions Loader concurrent program. When you install and set up Oracle Applications and Oracle Workflow, your system administrator must add this concurrent program to a request security group for the responsibility that you want to run this program from.

   See Also: *Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests*

2. Submit the Workflow Definitions Loader concurrent program as a request.

See Also: *Oracle Applications User's Guide, Submitting a Request*

3. In the Parameters window, specify Mode for the request:

- Specify *Download* of Mode to download a process definition from the database to a flat file.
- Specify *Upgrade* of Mode to apply a seed data upgrade to a database from an input file.
- Specify *Upload* of Mode to load a process definition from a flat file into the database.

4. Specify the full path and name of the file that you want to download a process definition to, or upgrade or upload a process definition from.

5. If you set Mode to *Download*, use the List button to choose the item type for the process definition to be downloaded.

Note: When you submit the Workflow Definitions Loader from the Submit Requests form to download process definitions to a file, you can only elect to download one item type at a time. To download multiple or all item types simultaneously, submit the Workflow Definitions Loader concurrent program from the command line.

See Also: *Oracle Workflow Administrator's Guide, Using the Workflow Definitions Loader*

Configuring the Business Event System

The CTB OMP XML definition file for the Business Event System (`ctbompe.wfx` – definition for Event, and `ctbomps.wfx` – definition for Event Subscription and Data) is located at: `ctb/src/patch/115/xml/US`

Use the Workflow XML Loader command line utility to upload and download the XML definitions for Business Event System objects between a database and a flat file (`ctbompe.wfx/ctbomps.wfx`). When uploading object definitions to a database, Oracle Workflow loads the definitions from the source XML file into the Business Event System tables in the database, creating new definitions or updating existing definitions as necessary. When downloading Business Event System object definitions from a database, Oracle Workflow saves the definitions as an XML file.

The XML definitions for Business Event System objects are structured in accordance with the following document type definitions (DTDs):

- **Event:** `WF_EVENTS DTD`, *Oracle Workflow API Reference*
- **Event group member:** `WF_EVENT_GROUPS DTD`, *Oracle Workflow API Reference*
- **Event subscription:** `WF_EVENT_SUBSCRIPTIONS DTD`, *Oracle Workflow API Reference*
- **System:** `WF_SYSTEMS DTD`, *Oracle Workflow API Reference*
- **Agent:** `WF_AGENTS DTD`, *Oracle Workflow API Reference*
- **Agent group member:** `WF_AGENT_GROUPS DTD`, *Oracle Workflow API Reference*

> **Note:** Agent groups are only available for the version of Oracle Workflow embedded in Oracle Applications.

To upload Business Event System object definitions from an XML file to a database, you can either run the Workflow XML Loader manually, or, if you are using the standalone version of Oracle Workflow, you can use a script to run the loader.

To run the Workflow XML Loader manually, run your Java Runtime Environment (JRE) against `oracle.apps.fnd.wf.WFXLoad`. You must specify your CLASSPATH pointing to the Java Runtime Environment, the directory containing the Workflow JAR files, the Oracle JDBC implementation, and the following Workflow JAR files:

- `wfjava.jar:` Workflow Java utilities
- `wfapi.jar:` Workflow Java APIs

> **Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, the Workflow JAR files are located in the `<ORACLE_HOME>/wf/java/oracle/apps/fnd/wf/jar/` directory.

For example, on UNIX, use the following command to run the Workflow XML Loader:

```
jre -classpath
"$<JREPATH>/rt.jar:$<Workflow_JAR_file_directory>:
$<Workflow_JAR_file_directory>/wfjava.jar:
$<Workflow_JAR_file_directory>/wfapi.jar:
$<ORACLE_HOME>/jdbc/lib/classes111.zip:"
oracle.apps.fnd.wf.WFXLoad -u[f] <user> <password>
<connect_string> <protocol> <lang> <source_file>
```

If you are using the standalone version of Oracle Workflow, you can use sample scripts called `wfxload` for UNIX or `wfxload.bat` for Windows NT to run the Workflow XML Loader. These scripts are located on a server in the Oracle Workflow *admin* subdirectory. For example, on UNIX, use the following command:

```
wfxload -u[f] <user> <password> <connect_string> <protocol> <lang> <source_file>
```

When running the Workflow XML Loader, use either the `-u` option or the `-uf` option to specify the upload mode that you want.

- `-u` —Normal upload mode. The Workflow XML Loader loads the object definitions from the source XML file into the Business Event System tables in the database, but does not make any updates to events or subscriptions with a customization level of **User**.
- `-uf` —Force upload mode. The Workflow XML Loader loads the object definitions from the source XML file into the Business Event System tables in the database and overwrites any existing definitions, including events or subscriptions with a customization level of *User*.

To download Business Event System object definitions from a database to a flat XML file, run the Workflow XML Loader with command:

```
jre -classpath
"$<JREPATH>/rt.jar:$<Workflow_JAR_file_directory>:
$<Workflow_JAR_file_directory>/wfjava.jar:
$<Workflow_JAR_file_directory>/wfapi.jar:
$<ORACLE_HOME>/jdbc/lib/classes111.zip:"
oracle.apps.fnd.wf.WFXLoad -d[e] <user> <password><connect_string> <protocol>
<lang> <output_file> <object> <key> OWNER_TAG <owner_tag>
```

For CTB OMP seeded BES definition, `<owner_tag>` is OMP.

> **See Also:** *Oracle Workflow Administrator's Guide, Using the Workflow XML Loader*

H

Empty Concept Lists

This Appendix contains the following reference table that documents predefined concept lists that contain no values—these lists must be populated with ETS concepts prior to using the associated functionality:

- Table H–1, Empty Concept Lists

See Also:

- Section 4.5.6.2, Adding Concepts to a Concept List
- *HTB Concept Lists Index*[1], *Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page),* for a list of all concept lists and their values.

[1] Lookup Types are synonymous with concept lists.

Table H–1 Empty Concept Lists

Subject Area	Concept List
Audit	CTB_AU_PURP
Clinical	CTB_CL_DIET_MODIFIER
Clinical (Knowledge Integration)	CTB_CL_KI_BMI
Clinical (Knowledge Integration)	CTB_CL_KI_BSA
Clinical (Knowledge Integration)	CTB_CL_KI_HEIGHT
Clinical (Knowledge Integration)	CTB_CL_KI_WEIGHT
Encounter Management	CTB_EM_ACUITY_LEVEL_CODE
Encounter Management	CTB_EM_ENC_TYPE_CODE
Encounter Management	CTB_EM_FINANCIAL_CLASS_CODE
Encounter Management	CTB_EM_GROUP_SUBTYPE_CODE
Encounter Management	CTB_EM_GROUP_TYPE_CODE
Encounter Management	CTB_EM_PUBLICITY_CODE
Equipment	CTB_EQ_FEATURE
Organization Setup	CTB_ORG_BUS_TYPE
Organization Setup	CTB_ORG_ENT_TYPE
Organization Setup	CTB_ORG_FAC_TYPE
Organization Setup	CTB_ORG_PS_CLASSFCTN
Patient Consent & Authorization	CTB_PCA_OPT_TYPE
Patient Consent & Authorization	CTB_PCA_PURP
Patient Registration	CTB_WORSHIP_PLACE
Scheduling	CTB_PS_PROC_CATEGORY
Security	CTB_SEC_PURP
Staff Management	CTB_SCP_ASSIGNMENT_CODE
Staff Management	CTB_SCP_CERTIFICATION_CODE
Staff Management	CTB_SCP_CLAIM_CODE
Staff Management	CTB_SCP_CLINICIAN_CODE
Staff Management	CTB_SCP_CRED_DOC_CODE
Staff Management	CTB_SCP_CREDENTIAL_CODE
Staff Management	CTB_SCP_EDUCATION_CODE
Staff Management	CTB_SCP_LICENSURE_CODE
Staff Management	CTB_SCP_POSITION_CODE

Table H–1 (Cont.) Empty Concept Lists

Subject Area	Concept List
Staff Management	CTB_SCP_PRIV_DEF_CODE
Staff Management	CTB_SCP_PRIVILEGE_CODE
Staff Management	CTB_SCP_REGISTRATION_CODE
Staff Management	CTB_SCP_SPECIALTY_CODE
Staff Management	CTB_SCP_STAFFMEMBER_CODE
Staff Management	CTB_SCP_TRAINING_CODE
Staff Management	CTB_SCP_VERIFICATION_CODE
Wait List Management	CTB_WL_ACTION_REASON

Concept List Equivalents

This Appendix contains the following reference table (Table I–1) that documents extensible concept lists that are the functional equivalent of FND lookups.

Note: If you extend any concept list or FND lookup listed in Table I–1, you must also extend its associated lookup type or concept list accordingly to keep them consistent.

See Also:

- Section 4.3.2.1, Organization Roles
- Section 4.5.6.2, Adding Concepts to a Concept List
- *HTB Concept Lists Index*[1]*, Oracle Javadoc for HTB (click HTB Concept Lists link at bottom of Javadoc page),* for a list of all concept lists and their values.

[1] Lookup Types are synonymous with concept lists.

Table I–1 Concept List and FND Lookup Equivalents

Concept List Name	FND Lookup Name
CTB_CONTACT_TITLE	FND_CONTACT_TITLE
CTB_LANGUAGE_CODE	FND_LANGUAGE
CTB_ORG_ROLE	FND_ORG_CLASS
CTB_PARTY_SITE_USE_CODE	FND_PARTY_SITE_USE_CODE
CTB_PHONE_LINE_TYPE	FND_PHONE_LINE_TYPE
CTB_REGISTRY_STATUS	FND_REGISTRY_STATUS

J

Party Merge Procedures

This appendix describes the Party Merge processes supported by HTB, including the process flow, details related to merged or transferred data, and any HTB validations that occur before completion of a process. This helps you determine:

- The impact of running the merge process.
- Changes required for party-related data.

This appendix includes the following sections:

Reference

- *Oracle Trading Community Architecture Party Merge User Guide*
- *Oracle Trading Community Architecture Data Quality Management User Guide*
- *Oracle Applications User Guide for 11i*

Background

Party Merge functionality is provided by Oracle Trading Community Architecture (TCA), which is used to resolve duplicates in the Trading Community registry.

HTB maintains additional data associated with person and organization parties, any references to which are updated when the parties are merged. This insures data integrity across the Oracle E-Business Suite. For organization parties, only external organization merge (Payer, Employer) is supported.

Process Flow

Party Merge can only be initiated between two active person or organization records. TCA supports the *survivor-donor* merge model, which is defined by the direction of the merge.

Party Merge can be initiated in one of the following ways:

- By using GUI-based TCA DQM Batch Match functionality (see *Oracle TCA Data Quality Management User Guide*, Chapter 5 for further details)
- By submitting a concurrent program (see *Oracle Applications User Guide for 11*, for further details)

When Person Party Merge or Organization Party Merge is initiated through TCA, HTB merge procedures are called to merge or transfer the relevant records in HTB extension tables. These procedures are written in PL/SQL and registered in the TCA Merge Dictionary, to be available for automatic calls after a merge process is finished for the TCA portion of the data.

After the merge is finished, the surviving party records remain active, while the corresponding donor records are marked as merged. The merged records are automatically filtered out from the data accessible through HTB APIs—these records cannot be queried, updated or cross-referenced.

The dependent data is either transferred or merged based upon additional validation checks as outlined below. The term *merge* as used herein refers to cases where a record exists in the surviving master party to which the record from the donor (duplicate) party is merged. For all merged entities, the values in the `record_status_code` column of the corresponding HTB tables are set to *M*

(merged). If the record belonging to the donor (duplicate) party now points to the surviving (master) party the record is considered *transferred*; the term *transfer* refers to the updates of foreign keys to point to the surviving party records.

Dependent merge entities include the following:

- Person records, including cross-reference data
- Insurance records
- Patient records
- Encounters records
- Clinical data
- Staff and clinician records
- External Organizations records

Person Merge Impact

The HTB merge process retains all information on the surviving records, and transfers as much information from the donor records as possible.

The following information *is always transferred* to the surviving master party:

- Cross-reference data
- Encounter and clinical records
- Contacts
- Privileges for staff assignments
- Patient list owners and filters
- Clinician specialties and credentials
- Workgroup and workgroup members information
- Staff participations and clinical act participations
- Wait list information

The following information is transferred *only if a duplicate record does not exist for the master party:*

- Person demographic information that allow for multiple values (names, identifications, races, ethnicities)
- Patient data, including MRN and medical conditions

- Patient insurance policies related information
- Related persons information
- PCP data
- Staff Member records, including staff assignments
- Staff member resources and resource group members
- Clinician records

Organization Merge Impact

Only external organizations of the type *Payer* and *Employer* can be merged. If you attempt to merge any internal organizations, or external organizations of a different type, an exception is raised and the merge does not take place (is vetoed).

The following information *is always transferred* to the surviving master party:

- Insurance policy eligibilities
- Encounter coverage, including authorization and copy information
- Payer and Plan contacts and locations
- Employer contacts

The following information is transferred *only if a duplicate record does not exist for the master party:*

- Employer related persons
- Payer insurance plans
- Insurance policies and policies' members

Party Merge Table Reference

Table J–1 lists the affected HTB tables, the associated transfer or merge action, and the HTB validations performed prior to completion of the party merge process:

Table J–1 Party Merge Table Reference

Tables affected	Action	Validation
CTB_PI_PARTIES	Transferred	For organization merges, the organization_type should be EXT_PAYER or EXT_EMPLOYER.
CTB_PI_PERSON_ CHRCTRSTCS	Conditional	If Person Characteristic code OR Characteristic Type code differs between the donor and the survivor records, transfer the donor record. Otherwise, merge.

Table J–1 (Cont.) Party Merge Table Reference

Tables affected	Action	Validation
CTB_PI_PARTY_IDNTFCTNS	Conditional	If the Identity Type code OR Issuing Organization ID differs between the donor and the survivor records, transfer the donor record. Otherwise, merge.
CTB_PI_PERSON_NAMES	Conditional	If both the donor and the survivor records have a preferred name, then the preferred name flag from the donor record is transferred with no preference flag set.
CTB_PI_CROSS_REF	Transferred	None
CTB_PR_INS_POLICIES	Conditional	If policy_num, org_party_id, insurance_plan_id, group_name, group_num, effective_start_date, and effective_end_date of donor patient are same as surviving patient then that Policy is merged. Otherwise, it is transferred.
CTB_PR_INS_POL_MEMBERS	Conditional	If insurance_policy_id, effective_start_date, and effective_end_date of donor person's Policy member are same as that of the survivor person's Policy member, then Policy member is merged. Otherwise, it is transferred.
CTB_PR_MEDICAL_CONDS	Conditional	If medical_condition_type_code and medical_ condition_code of donor person is present for the survivor person, then record is merged. Otherwise, it is transferred.
CTB_PR_PATIENTS	Conditional	If survivor patient is from the same organization as the donor patient, then the patient is merged. Otherwise, it is transferred.
CTB_PR_RELATED_PERSONS	Conditional	If donor's related person's related_person_type_code, relation_code, first_name, middle_name and last_ name are same as surviving person's related person then that related person is merged. Otherwise, it is transferred.
CTB_PR_RELATED_PERSONS	Conditional	If donor organization's related person's related_ person_type_code, relation_code, first_name, middle_ name and last_name are same as surviving organization's related person's then that related person is merged. Otherwise, it is transferred.
CTB_PR_PCPS *(triggered by Patient Person Merge)*	Conditional	If donor patient's registered PCP's provider_person_ party_id, effective_start_date, and effective_end_date are same as the surviving patient's registered PCP's then that PCP is merged. Otherwise, it is transferred.If donor patient's non-registered PCP's n.
CTB_PR_PCPS *(triggered by Provider Person Merge)*	Conditional	If donor's PCP's provider_person_party_id, effective_ start_date, and effective_end_date are same as surviving PCP's then that PCP is merged. Otherwise, it is transferred.

Table J–1 (Cont.) Party Merge Table Reference

Tables affected	Action	Validation
CTB_PR_INS_POLICIES	Conditional	If policy_num, org_party_id, insurance_plan_id, group_name, group_num, effective_start_date, and effective_end_date of donor's organization are same as the surviving organization then that Policy is merged. Otherwise, it is transferred.
CTB_PR_INS_PLANS	Conditional	If donor organization Plan's plan_long_name or plan_short_name are same as surviving organization Plan's then that Plan is merged. Otherwise, it is transferred.
CTB_PR_MASTER_CONTACTS *(triggered by Organization's address merge)*	Transferred	None
CTB_PR_MASTER_CONTACTS *(triggered by External organization merge)*	Transferred	None
CTB_SR_ORG_STAFF_MEMBERS	Conditional	If donor staff member is from the same organization as the surviving staff member, then the staff member is merged. Otherwise, it is transferred.
CTB_EM_ENC_IDS	Transferred	None
CTB_SR_STAFF_MEMBERS	Conditional	If donor staff member is scoped by the same organization unit as the surviving staff member, AND the donor staff member has the same status as the surviving staff member or status of *Nullified*, the staff member record is merged; otherwise, it is transferred.
CTB_SR_CLINICIANS	Conditional	If both donor and surviving person have an associated clinician record the donor clinician is merged; otherwise it is transferred.

HTB Party Merge Procedures

Table J–2 lists the HTB procedures that are registered in the TCA Merge Dictionary:

Table J–2 HTB Party Merge Procedures

Tables Affected	Procedure Name
CTB_PI_PARTIES	CTB_EMPI_PKG.Merge_Parties
CTB_PI_PERSON_CHRCTRSTCS	CTB_EMPI_PKG.Merge_Person_Char
CTB_PI_PARTY_IDNTFCTNS	CTB_EMPI_PKG.Merge_Party_Ident
CTB_PI_PERSON_NAMES	CTB_EMPI_PKG.Merge_Person_Names
CTB_PI_CROSS_REF	CTB_EMPI_PKG.Merge_Cross_Ref

Table J–2 (Cont.) HTB Party Merge Procedures

Tables Affected	Procedure Name
CTB_PR_INS_POLICIES	CTB_PR_Patient_Merge_Pvt.Merge_Ins_Policies1
CTB_PR_INS_POL_MEMBERS	CTB_PR_Patient_Merge_Pvt.Merge_Ins_Pol_Members2
CTB_PR_MEDICAL_CONDS	CTB_PR_Patient_Merge_Pvt.Merge_Medical_ Conditions
CTB_PR_PATIENTS	CTB_PR_Patient_Merge_Pvt. Merge_Patients
CTB_PR_RELATED_PERSONS	CTB_PR_Patient_Merge_Pvt. Merge_Related_Persons
CTB_PR_RELATED_PERSONS	CTB_PR_Patient_Merge_Pvt. Merge_Related_ Persons2
CTB_PR_PCPS	CTB_PR_Patient_Merge_Pvt. Merge_Pcps1
CTB_PR_PCPS	CTB_PR_Patient_Merge_Pvt. Merge_Pcps2
CTB_PR_INS_POLICIES	CTB_PR_Patient_Merge_Pvt. Merge_Ins_Policies3
CTB_PR_INS_PLANS	CTB_CS_EXT_ORG_MERGE_PUB.Merge_Insurance_Plans
CTB_PR_MASTER_CONTACTS	CTB_CS_EXT_ORG_MERGE_PUB.Merge_Ext_Org_Contacts
CTB_PR_MASTER_CONTACTS	CTB_CS_EXT_ORG_MERGE_PUB.Merge_Contact_Party_ Sites
CTB_SR_ORG_STAFF_MEMBERS	CTB_SR_Staff_Merge_Pvt. Merge_Staff_Members
CTB_EM_ENC_IDS	CTB_EM_Encounter_Merge_Pvt. Merge_Person_ Encounters
CTB_SR_STAFF_MEMBERS	CTB_SCP_Staffmgmt_Merge_Pvt.Merge_StaffMembers
CTB_SR_CLINICIANS	CTB_SCP_Staffmgmt_Merge_Pvt.Merge_Clinicians

K

Act Definition Messaging Considerations

This appendix describes act definition messaging considerations, including proposed act definitions that can be used in each message domain.

Inbound Messaging Services processes inbound clinical and administrative HL7 messages—one of the two mechanisms used to populate patient data into HTB. Both Inbound Messaging Services and applications developed with HTB use its Applications Programming Interface to create and update data in the HTB repository. HTB Inbound Messaging services uses **Act Definition**s for the processing and persisting of clinical data. The following section briefly describes the types of messages that can be passed into HTB and those that require the use of Act Definitions. An act definition consists of class, mood, category code, `detailID` type and `detailId`. The examples that follow in this appendix use the following notation to describe an act definition:

```
Class.Mood.CategoryCode.DetailTypeID.detailID
```

The `detailID` is an ETS ID that is derived from a concept code and coding scheme name that is sent in a message (but is not displayed in the examples). Typically, when an act definition includes a `detailID`, there are many act definitions represented in that message domain.

This appendix refers to message domains and act relationships defined in the HTB HL7 Version 3 Conformance Specifications (the Conformance Specifications).

> **See Also:** *Oracle Healthcare Transaction Base HL7 Version 3 Conformance Specifications, Release 11i—Patch Set D,* available on *OracleMetaLink.*
>
> Note that there are two specifications—one for Inbound Messaging Services, and one for Outbound Messaging Services.

Table K–1 lists message domains described by this appendix, and indicates which domains use act definitions:

Table K–1 Message Domains

Message Domain Name	Uses Act Definitions?
Bed Status Observation Event	No
Condition Problems	Yes
Diagnostic Report Observation Event	Yes
Diet Request (Order)	Yes
Encounter Appointment	Yes
Encounter Event	Yes

Table K–1 (Cont.) Message Domains

Message Domain Name	Uses Act Definitions?
Intolerance Observation Event	Yes
Medication Supply Events	Yes
Observation Event	Yes
Observation Order	Yes
Person Merge	No
Person Registry	No
Procedure Orders	Yes
Specimen Observation Event	Yes
Specimen Observation Order	Yes
Staff Registry	No
Substance Administration Event	Yes
Substance Administration Order	Yes
Supply Request (Order)	Yes

The following sections describe proposed act definitions that can be used in each message domain:

- Condition Problems
- Diagnostic Report Observation Event
- Diet Request (Order)
- Encounter Appointment
- Encounter Event
- Intolerance Observation Event
- Medication Supply Events
- Observation Event
- Observation Order
- Procedure Orders
- Specimen Observation Event

- Specimen Observation Order
- Substance Administration Event
- Substance Administration Order
- Supply Request (Order)

Condition Problems

Condition Problems have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation`
- `reason`

Example K–1 Sample Act Definitions:

- `Focal` act: COND.EVN.PROBLEM.NULL
- `pertinentInformation` act: OBS.EVN.NOTE.NULL – this is an annotation
- reason act: OBS.EVN.REASON.NULL or,
- OBS.EVN.INTDX.NULL, etc.

Diagnostic Report Observation Event

Diagnostic Report Observation Event has an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation1`
- `pertinentInformation2`
- `inFulfillmentOf`
- `replacementOf`
- `appendage`
- `documentationOf`
- `reason`

Example K–2 Sample Act Definitions:

- `Focal` act: OBS.EVN.DXIMG.ANY – This act definition indicates if it is a chest x-ray or an EKG or any other diagnostic image. This is where the pointer to the image can optionally be stored when the act is created.
- `PertinentInformation1` act: Many different types of act definitions can be seen here, including OBS.EVN.LAB.ANY, or, OBS.EVN.SS.ANY, or OBS.EVN.(site defined category code). NULL
- `PertinentInformation2` act: OBS.EVN.NOTE.NULL
- `inFulfillmentOf` act: OBS.RQO.DXIMG.ID or PROC.RQO.DXIMG.ID
- `replacementOf` act: OBS.EVN.DXIMG.ANY
- `appendage` act: OBS.EVN.DXIMG.ANY, other possibilities, OBS.EVN.NOTE.NULL
- `documentationOf` act: OBS.EVN.REPORT.NULL, or OBS.EVN.RADIOL.NULL, or OBS.EVN.CARDIO.NULL, etc. This act definition is where the text of the report would be sent. You need the focal act to understand what the report is for (chest x-ray, etc.).
- `reason` act: OBS.EVN.REASON.NULL or, OBS.EVN.INTDX.NULL, etc.

Diet Request (Order)

Diet Request (Order) has an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation1`
- `pertinentInformation2`
- `reason`

Example K–3 Sample Act Definitions:

- `Focal` act: DIET.RQO.MEAL.NULL, or DIET.RQO.SNACK.NULL, or DIET.RQO.SUPPLEMENT.NULL
- `PertinentInformation1` act: SBADM.RQO.TUBE.ANY or SBADM.RQO.TUBE.ID —- optionally used to indicate the rate of a tube feeding dietary supplement
- `PertinentInformation2` act: OBS.EVN.NOTE.NULL

- `reason` act: OBS.EVN.REASON.NULL or, OBS.EVN.INTDX.NULL , COND.EVN.PROBLEM.NULL etc.

Encounter Appointment

Encounter Appointment has an act definition for each act represented in the following act relationships:

- `pertinentInformation1`
- `pertinentInformation3`
- `reason`

Example K–4 Sample Act Definitions:

- `pertinentInformation1` act: Many different types of act definitions can be seen here, including: OBS.EVN.LAB.ANY, or, OBS.EVN.SS.ANY, or OBS.EVN.(site defined category code). NULL
- `pertinentInformation3` act: OBS.EVN.NOTE.NULL
- `reason` act: OBS.EVN.REASON.NULL or, OBS.EVN.INTDX.NULL, etc.

Encounter Event

Encounter Event has an act definition for each act represented in the following act relationships:

- `pertinentInformation3`
- `pertinentInformation4`
- `reason`

Example K–5 Sample Act Definitions:

- `pertinentInformation3` act: The following act definitions are required to support administrative observations: OBS.EVN. DIET_PREF.NULL, or OBS.EVN.EXPECT_SURG.NULL, or OBS.EVN.EXPECT_DSCH, or OBS.EVN.REL_CLIN_INFO.NULL
- `pertinentInformation4` act: OBS.EVN.INTDX.NULL, or OBS.EVN.ADMDX.NULL, or OBS.EVN.DISDX.NULL
- `reason` act: OBS.EVN.REASON.NULL, or OBS.EVN.PRI_REASON.NULL, or OBS.EVN.GEN_REASON.NULL, or OBS.EVN.ADMT_REASON, or OBS.EVN.TRANSF_REASON, etc.

Intolerance Observation Event

Intolerance Observation Event has an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation1`
- `pertinentInformation2`
- `pertinentInformation3`
- `causeOf`

Example K–6 Sample Act Definitions:

- `Focal` act: OBS.EVN.ALL.NULL
- `pertinentInformation1` act: many different types of act definitions can be seen here, including: OBS.EVN.LAB.ANY, or, OBS.EVN.SS.ANY, or OBS.EVN.(site defined category code). NULL
- **`pertinentInformation2`** act: OBS.EVN.SEV.NULL
- `pertinentInformation3` act: OBS.EVN.NOTE.NULL
- `causeOf` act: OBS.EVN.ADVERSE_REACT.NULL

Medication Supply Events

Medication Supply Events have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation`
- `inFulfillmentOf`

Example K–7 Sample Act Definitions:

- `Focal` act: SPLY.EVN.CLIN_DRUG.NULL, etc.
- `pertinentInformation` act: OBS.EVN.NOTE.NULL
- `inFulfillmentOf` act: SBADM.RQO.CLIN_DRUG.ID or SBADM.RQO.CLIN_DRUG.ANY, etc.

Observation Event

Observation Events have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- component
- pertinentInformation1
- pertinentInformation2
- inFulfillmentOf
- reason

Note that observation reference ranges are stored as attributes of the focal act and not as separate clinical acts.

Example K–8 Sample Act Definitions:

- Focal act: Many different types of act definitions can be seen here, including: OBS.EVN.SS.ANY, or OBS.EVN.(category code here).ID, etc.
- component act: many different types of act definitions can be seen here, including: OBS.EVN.SS.ANY, or OBS,EVN.(category code here).ID, etc.
- pertinentInformation1 act: many different types of act definitions can be seen here, including: OBS.EVN.LAB.ANY, or, OBS.EVN.SS.ANY, or OBS.EVN.LAB.ID, OBS.EVN.(category code here).ID, etc.
- inFulfillmentOf act: OBS.RQO.SS.ID, or OBS.RQO.(category code here).ID, etc.
- reason act: OBS.EVN.REASON.NULL

Observation Order

Observation Order have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- pertinentInformation
- occurrenceOf
- precondition
- reason
- risk
- support

Example K–9 Sample Act Definitions:

- `Focal` act: Many different types of act definitions can be seen here, including OBS.RQO.SS.ID, etc.
- `pertinentInformation` act: OBS.EVN.NOTE.NULL
- `occurrenceOf` act: many different types of act definitions can be seen here, including OBS.RQO.SS.ID, etc.
- `precondition` act: OBS.EVN.CRT.PRECONDITION.NULL
- `reason` act: OBS.EVN.REASON.NULL or, OBS.EVN.INTDX.NULL, COND.EVN.PROBLEM.NULL etc.
- `risk` act: OBS.EVN.(create category code - RISK).NULL
- `support` act: Many different types of act definitions can be seen here, including OBS.EVN.LAB.ANY, or OBS.EVN.SS.ANY

Procedure Orders

Procedure Orders have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation`
- `occurrenceOf`
- `precondition`
- `reason`
- `risk`
- `support`

Example K–10 Sample Act Definitions:

- `Focal` act: many different types of act definitions can be seen here, including: PROC.RQO.(category code here).ID, etc.
- `pertinentInformation` act: OBS.EVN.NOTE.NULL
- `occurrenceOf` act: many different types of act definitions can be seen here, including PROC.RQO.(category code here).ID, etc.
- `precondition` act: OBS.EVN.CRT.PRECONDITION.NULL
- `reason` act: OBS.EVN.REASON.NULL or, OBS.EVN.INTDX.NULL, COND.EVN.PROBLEM.NULL etc.

- `risk` act: OBS.EVN.(create category code - RISK).NULL
- `support` act: many different types of act definitions can be seen here, including OBS.EVN.LAB.ANY, or OBS.EVN.SS.ANY

Specimen Observation Event

Specimen Observation Events have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `component`
- `pertinentInformation1`
- `pertinentInformation2`
- `inFulfillmentOf`
- `reason`

Note that observation reference ranges are stored as attributes of the focal act and not as separate clinical acts.

Example K–11 Sample Act Definitions:

- `Focal` act: Many different types of act definitions can be seen here, including: OBS.EVN.LAB.ANY, or OBS.EVN.LAB.ID
- `component` act: OBS.EVN.LAB.ANY, or OBS.EVN.LAB.ID
- `pertinentInformation1` act: many different types of act definitions can be seen here, including: OBS.EVN.LAB.ANY, or, OBS.EVN.SS.ANY, or OBS.EVN.LAB.ID, OBS.EVN.(category code).ID, etc.
- `inFulfillmentOf` act: OBS.RQO.LAB.ID
- `reason` act: OBS.EVN.REASON.NULL

Specimen Observation Order

Specimen Observation Order have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation`
- `occurrenceOf`
- `precondition`
- `reason`

- `risk`
- `support`

Example K–12 Sample Act Definitions:

- `Focal` act: Many different types of act definitions can be seen here, including: OBS.RQO.LAB.ID, or OBS.RQO.LAB.ANY
- `pertinentInformation` act: OBS.EVN.NOTE.NULL
- `occurrenceOf` act: OBS.RQO.LAB.ID, or OBS.RQO.LAB.ANY
- `precondition` act: OBS.EVN.CRT.PRECONDITION.NULL
- `reason` act: OBS.EVN.REASON.NULL, or COND.EVN.PROBLEM.NULL, etc.
- `risk` act: OBS.EVN.(create category code - RISK).NULL
- `support` act: many different types of act definitions can be seen here, including OBS.EVN.LAB.ANY, or OBS.EVN.SS.ANY

Substance Administration Event

Substance Administration Events have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation`
- `inFulfillmentOf`
- `reason`

Example K–13 Sample Act Definitions:

- `Focal` act: Many different types of act definitions can be seen here, including: SBADM.EVN.CLIN_DRUG.ANY, or SBADM.EVN.IMMUN.ANY, or SBADM.EVN.MED_HX.ANY, or SBADM.EVN.(category code here).ID, etc.
- `pertinentInformation` act: OBS.EVN.NOTE.NULL
- `inFulfillmentOf` act: SBADM.RQO.CLIN_DRUG.ID, or SBADM.RQO.IMMUN.ID, or SBADM.RQO.(category code here).ID, etc.
- `reason` act: OBS.EVN.REASON.NULL

Substance Administration Order

Substance Administration Orders have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `component`
- `pertinentInformation`
- `occurrenceOf`
- `precondition`
- `predecessor`
- `reason`

Example K–14 Sample Act Definitions:

- `Focal` act: Many different types of act definitions can be seen here, including: SBADM.RQO.CLIN_DRUG.ID, or SBADM.RQO.IMMUN.ID, or SBADM.RQO.(category code here).ID, etc.
- `component` act: many different types of act definitions can be seen here, including: SPLY.RQO.CLIN_DRUG.ID, or SPLY.RQO.IMMUN.ID, etc.
- `pertinentInformation1` act: OBS.EVN.NOTE.NULL
- `pertinentInformation2` act: many different act definitions can be used here, including: OBS.EVN.SS.ID, or OBS.EVN.SS.ANY, etc.
- `precondition` act: OBS.EVN.CRT.PRECONDITION.NULL
- `predecessor` act: many different types of act definitions can be seen here, including: SBADM.RQO.CLIN_DRUG.ID, or SBADM.RQO.IMMUN.ID, or SBADM.RQO.(category code here).ID, etc.
- `reason` act: OBS.EVN.REASON.NULL or, OBS.EVN.INTDX.NULL, COND.EVN.PROBLEM.NULL etc.

Note: Substance Administration Orders also have act definitions created to support complex intravenous orders such as Total Parental Nutrition (TPN) orders. These act definitions are: SBADM.RQO.BASE_CLIN_DRUG.ID, and SBADM.RQO.ADD_CLIN_DRUG.ID

Supply Request (Order)

Supply Requests have an act definition for the focal act in the message and an act definition for each act represented in the following act relationships:

- `pertinentInformation`
- `reason`

Example K–15 Sample Act Definitions:

- `Focal` act: Many different act definitions can be seen here, including: SPLY.RQO.SUPPLY.ID, or SPLY.RQO.SUPPLY.ANY, or SPLY.RQO.CLIN_DRUG.ID, or SPLY.RQO.IMMUN.ID, etc.
- `reason` act: OBS.EVN.REASON.NULL, or OBS.EVN.INTDX.NULL, or COND.EVN.PROBLEM.NULL etc.

L

Clinical Business Services and ETS Concept Lists

This appendix describes the use of ETS concept lists in Clinical Business Services.

Enterprise Terminology Services (ETS) plays a vital role in HTB by providing support for loading standard and local terminologies. When implementing Clinical Business Services, it is important to carefully consider which terminology or terminologies are loaded into ETS in order to support the recording of Clinical Acts. The terminologies that are used can be a standard terminology, a local terminology or some combination of local and standard terminologies.

Mapping a local terminology to a standard terminology or using the standard terminology directly is preferable to using a local terminology alone or using it without mapping it to a standard terminology. Mapping to a standard terminology provides a common conceptual basis for all clinical data within the enterprise. It also facilitates clinical data exchange with other internal and external systems.

Clinical Business Services principally uses concepts from ETS to support clinical services and master catalog services:

- **Clinical Services:** Coded attributes in Clinical Services either store an ETS ID or a membership code in HTB. In both cases ETS concepts are either directly referenced with an ETS ID, or indirectly referenced via a membership code. The Java *set* methods for these attributes validate coded values as either a valid ETS concept, or as a valid membership code in a specific pre-loaded concept list.
- **Master Catalog Services:** Coded attributes in Master Catalog Services either store an ETS ID or a membership code in HTB. In both cases ETS concepts are either directly referenced with an ETS ID, or indirectly referenced via a membership Code. The Java *set* methods for these attributes validate coded values as either a valid ETS concept, or as a valid membership code in a specific pre-loaded concept list. To create Act Definitions for the master catalog, the user must populate the following coded attributes: Class, Mood, Category Code and Detail ID (optional). In HTB the class, mood and category code attributes are stored as membership codes from their respective concept lists. The `DetailId` attribute is stored as an ETS ID in HTB.

Concept Lists consist of valid ETS concepts and are used to assure that any values associated with an attribute are valid. Attributes that refer to concept lists for validation are populated with membership codes from that concept list. The customer must evaluate the extensible concept lists and determine if additional values need to be added to the list. For information about what values are contained in each concept list, see the clinical attribute concept list link in the HTB Javadoc.

See Also:

- *Oracle Javadoc for HTB* for information about concept lists and their contents
- Section 4.5.6, Implementing Concept Lists, for information about ETS concept lists

Table L–1 describes Clinical Business Service and Master Catalog Service attributes that use concept lists:

Table L–1 Clinical Business Service Attributes that use Concept Lists

Class	Attribute	Concept List Type[1]	Concept List Populated?	Task[2]
Act Definition	`ClassCode`	SYSTEM	Y	NA
Act Definition	`MoodCode`	SYSTEM	Y	NA
Act Definition	`CategoryCode`	EXTENSIBLE	Y	Evaluate/add
Act Definition	`ConfidentialityCode`	EXTENSIBLE	Y	Evaluate/add
Act Participation	`ContextControlCode`	SYSTEM	Y	NA
Act Participation	`ModeCode`	EXTENSIBLE	Y	Evaluate/add
Act Participation	`SubstitutionConditionCode`	EXTENSIBLE	N	Evaluate/add
Act Relationship	`CheckpointCode`	SYSTEM	Y	NA
Act Relationship	`ConjunctionCode`	SYSTEM	Y	NA
Act Relationship	`RelationshipTypeCode`	SYSTEM	Y	NA
Act Type	`Class`	SYSTEM	Y	NA
Act Type	`Mood`	SYSTEM	Y	NA
Clinical Act	`UncertaintyCode`	SYSTEM	Y	NA
Clinical Act	`StatusCode`	SYSTEM	Y	NA
Clinical Act	`HL7ClassCode`	SYSTEM	Y	NA
Clinical Act	`HL7MoodCode`	SYSTEM	Y	NA
Clinical Act	`CategoryCode`	EXTENSIBLE	Y	See Act Definition
Clinical Act	`HL7ConfidentialityCodes`	EXTENSIBLE	Y	See Act Definition
DietAct	`DietModifiers`	EXTENSIBLE	N	Evaluate/add

Table L–1 (Cont.) Clinical Business Service Attributes that use Concept Lists

Class	Attribute	Concept List Type[1]	Concept List Populated?	Task[2]
Material	`CapTypeCode`	EXTENSIBLE	Y	Evaluate/add
Material	`HandlingCode`	EXTENSIBLE	Y	Evaluate/add
Material	`RiskCode`	SYSTEM	Y	NA
Material	`EntityClassCode`	SYSTEM	Y	NA
Material Participation	`TypeCode`	SYSTEM	Y	NA
Material Participation	`SubstitutionConditionCode`	EXTENSIBLE	N	Evaluate/add

[1] There are two types of concept lists: (i) EXTENSIBLE (concepts can be added to the list), and (ii) SYSTEM (not EXTENSIBLE).

[2] This column identifies those concepts that should be evaluated for potential addition values (to be added).

See Also: The following White Papers, available on *OracleMetaLink*:

- *Using Enterprise Terminology Services in HTB to Build Healthcare Applications*
- *Implementing Terminologies in the Generic ETS Model*

M

Running Concurrent Programs

This appendix describes how to run Oracle Concurrent programs using the Oracle Concurrent Manager. The procedure is the same for all Oracle Applications.

Steps

1. Log in as SYSADMIN.
2. Select Systems Administrator responsibility.
3. Double-click **Concurrent Requests.**
4. Double-click **Run**.
5. Click the Single Request button.
6. Click **OK**.
7. Search for the target concurrent request.
8. Click **Submit**.
9. Select from the search results.
10. Click **OK**.
11. Click **OK** to run the concurrent request.

> **See Also:** *Oracle Applications System Administrator's Guide, Overview of Concurrent Programs and Requests,* for more information about the Oracle Concurrent Manager

N

Abbreviations & Acronyms

This appendix defines abbreviations and acronyms used in the Oracle Healthcare Transaction Base Implementation Guide (Table N–1):

Table N–1 Abbreviations and Acronyms

Abbreviation / Acronym	Description
ABG	Arterial blood gasses
ADT	Admit, discharge and transfer
AGS	Administrative grouping of services; encounter group
AMA	American Medical Association
API	Applications Programming Interface
B2B	Business to Business
B2C	Business to Customer
CA	Certificate Authority
CAT	Computer-Assisted Tomography; See also: CT
CBC	Complete blood count
CDT	Current Dental Terminology
CDT-2	Current Dental Terminology, 2nd Revision
CLIA	Clinical Laboratories Improvement Act
CMS	Centers for Medicare & Medicaid Services; was HCFA
CPT	Current Procedural Terminology
CPT4	Current Procedural Terminology, 4th Revision
CT	Computerized Tomography; See also: CAT
CTB	Oracle Clinical Transaction Base; now HTB
DES	Data Encryption Standard (U.S.)
DHHS	U.S. Department of Health and Human Services
DMIM	Domain Message Information Model
DN	Distinguished Name
DNS	Domain Naming Service
DRG	US Diagnosis Related Group
DRS	Designated Record Set
E&M	Evaluation and Management Guidelines
ECG	Electrocardiogram; electrocardiography

Table N–1 (Cont.) Abbreviations and Acronyms

Abbreviation / Acronym	Description
EEG	Electroencephalogram; Electroencephalography
EHR	Electronic health record
EMPI	Enterprise Master Person Index; now called Person Services
EMS	Emergency Medical Services
EPS	Enterprise Patient Scheduling
ER	Emergency Room
ETS	Oracle Enterprise Terminology Services
ETSID	An internal identifier for an ETS entity or structure.
FDB	First Data Bank
GTS	General Timing Specification
GUI	Graphic User Interface
HCFA	Health Care Financing Administration; now CMS
HCPCS	Healthcare Financing Administration Common Procedural Coding System
HCPCS Level II	Healthcare Financing Administration Common Procedural Coding System, Level II
HCSM	Oracle Healthcare Staff Management
HHS	U.S. Department of Health and Human Services
HIPAA	Health Insurance Portability and Accountability Act of 1996
HL7	Health Level 7
HMD	Hierarchical Message Description
HPI	History of present illness
HR	Human Relations; relates to Oracle Human Resource Management, a proprietary applications software product of Oracle Corporation.
HRMS	Oracle Human Resource Management, a proprietary applications software product of Oracle Corporation.
HSS	U.S. Department of Health and Human Services
HTB	Oracle Healthcare Transaction Base; was CTB

Table N–1 (Cont.) Abbreviations and Acronyms

Abbreviation / Acronym	Description
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP combined with underlying SSL layer
ICD-9-CM	International Classification of Diseases—9th Revision, Clinical Modification
ICD-10	International Statistical Classification of Diseases and Health-related problems, 10th Revision
ICU	Intensive Care Unit
IETF	Internet Engineering Task Force
IETF RFC 1766	Internet Engineering Task Force Request for Comments 1766: Tags for the Identification of Languages
II	Instance Identifier
ISO	International Standards Organization
ISO 3166-1	International Standards Organization 3166-1: Country Codes
ISP	Internet Service Provider
ISV	Independent Software Vendor; Independent Service Vendor
JAR	Java archive file; contains compressed Java classes
JDBC	Java Database Connectivity
JDK	Java Development Kit
JRE	Java Runtime Environment
JSP	Java Server Pages
JVM	Java Virtual Machine
LAN	Local Area Network
LDS	Limited Data Set
LOINC	Logical Observation Identifier of Names and Codes
LOV	List of Values
MDC	US Major Diagnostic Categories
MDF	Message Development Framework (RIM)

Table N–1 (Cont.) Abbreviations and Acronyms

Abbreviation / Acronym	Description
MRI	Magnetic Resonance Imaging; Medical Records Information
MRN	Medical Record Number
MT	Message Type
NDDF	US National Drug Data File
OHCA	Organized Healthcare Arrangements
OID	Object Identifier; Oracle Internet Directory
PCP	Primary Care Provider; Primary Care Physician
PE	Physical Exam
PHI	Personal Health Information
PIN	Personal Identification Number
PKE	Public Key Encoding
PKI	Public Key Infrastructure
RIM	Reference Information Model (HL7)
RMIM	Refined Message Information Model
SCHIP	State Children's Health Insurance Program (U.S.)
SDO	Standards Developing Organization
SNOMED	Systematized Nomenclature of Medicine
SNOMED CT	Systematized Nomenclature of Medicine Clinical Terms
SQL	Structured Query Language
SSL	Secure Sockets Layer
SSO	Single Sign-on
TCP/IP	Transmission Control Protocol / Internet Protocol
TPO	Treatment, Payment, or Healthcare Operation
UB92	Universal Billing Document [1992]
UML	Unified Modeling Language
USAM	Unified Service Action Model
WAN	Wide Area Network

Table N–1 (Cont.) Abbreviations and Acronyms

Abbreviation / Acronym	Description
XML	Extensible Markup Language

Glossary

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

accompanying person

The person accompanying a **patient** to or during an **encounter**.

Act Definition

An ETS **concept** that has been added to the **master catalog**. Act definitions include class, **mood**, category code, detail ID type and detail ID.

act

action

See: **clinical act**.

Act Relationship

An association between a pair of acts. This includes act-to-act associations such as collector/component, predecessor/successor, and cause/outcome. An HL7 version 3 concept.

Act Type

The intersection of an act mood and an act class, used to define logical groupings of clinical act definitions.

admission

The arrival of a new **inpatient**. May follow **pending admission** or optional **pre-admission**.

admitting physician

The physician who authorizes a patient admission; a physician authorized to admit patients.

ambulatory encounter

An **encounter** that does not result in the hospitalization or institutionalization of the patient in the facility providing the service.

appointment

A time period scheduled for a patient **encounter**. It must specify the start date and time, end date and time, **encounter class**, and organization unit.

appointment status

The current state of an **appointment**. Valid statuses include reserved, active, aborted, nullified, and completed.

appointment type

The nature of the **appointment** (length, description, resource requirements) and the actions to be performed. This can also include associated diagnostic procedures. Examples include allergy shots, follow-up consultations, chemotherapy. Specific appointment types are assigned to one or more **resource**s or resource groups.

attending physician

The physician who has primary responsibility for the care of a specific patient for the duration of an encounter. See also: **primary care provider** (PCP).

authentication

The process of verifying the identity of a user, device, or other entity in a computer system or network, typically as a prerequisite to granting access to resources in a system. A recipient of an authenticated message can be certain of the message's origin (its sender). Authentication is presumed to preclude the possibility that another party has impersonated the party being authenticated.

authorization

Permission given to a user, program, or process to access an object or set of objects. The set of privileges available to an authenticated user or entity. See also: **patient authorization**.

B

business unit

A division or department of a healthcare **facility** or **enterprise** that supports an administrative, non-clinical function, such as a division or a department.

C

care site

Specific location of an **encounter** within a healthcare facility, such as a bed or room. There can be multiple care sites within a **practice setting**, and multiple **practice setting**s can use the same care site. Multiple patients can also be assigned to the same care site.

Centers for Medicare & Medicaid Services

Was the **Health Care Financing Administration** (HCFA). An agency of the **U.S. Department of Health and Human Services**, HCFA administers Medicare, Medicaid, and SCHIP. The agency also performs a number of quality-focused activities, including regulation of laboratory testing (CLIA) and development of coverage policies.

certification

A method of formally identifying healthcare **practitioner**s who have completed specified training, a certain set of requirements, or passed an examination. The certifying organization may be an academic, governmental, or professional body within the healthcare community.

ciphertext

Text (or a message) that has been encrypted.

clinical act

An instance of a clinical event; any clinical observation, service, procedure, or supply rendered as part of the diagnosis or treatment of a **patient**.

clinical assignment

The assignment of a **clinician** to a staff position for purposes of defining individual clinical responsibilities or **privileges** within an enterprise.

clinical attribute

A characteristic of a **clinical act** that provides descriptive information; an attribute associated with a **clinical act**.

clinical position

A clinical role within an enterprise.

clinical view

A collection of clinical data defined for rapid retrieval, for purposes of display or data entry.

clinician

See: **practitioner**.

coding

A mechanism for identifying and defining physician and hospital services, coding is a function of billing. Coding provides universal definition and recognition of diagnoses, procedures and level of care. A national certification exists for coding professionals and many compliance programs are currently raising standards of quality for their coding procedures.

coding scheme

Within ETS, a particular structured system of terms or concepts used to maintain coded meanings. Examples include The International Classification of Diseases, 9th Edition, Clinical Modification (ICD-9-CM), and University Hospital's Laboratory Codes.

coding scheme version

A particular instance of a **coding scheme**. For example: ICD-9-CM for the year 2000; ICD-9-CM for the year 2001; the University Hospital's Laboratory Codes, Updated September 2001.

compliance

Consistently and accurately conforming to U.S. government rules for Medicare billing system requirements and other regulations. A compliance program is a self-monitoring system of checks and balances intended to ensure that an

organization consistently complies with applicable laws relating to its business activities or healthcare delivery services.

concept

Cognitive construct (or abstraction) formed by using the characteristics of objects; unit of thought. A concept should not be confused with a linguistic or symbolic scheme used to represent it. Objects can be thought of as instances of concepts. Example: the concept of viral hepatitis.

concept code

Identifies a **concept** within its native terminology.

ConceptID

An internal identifier for an ETS **concept**.

concept list

Within ETS, user-defined groupings of ETS concepts that can be used by other applications; for example, lookup types are ETS concepts.

See: *Oracle Javadoc Lookup Type Index (concept lists)* for a list of seeded concept lists and values. These concept lists represent the valid sets of coded values used by HTB APIs and by HTB message processing.

consent

Permission given to **provider**s by individuals for the use and disclosure of **protected health information** for treatment, payment, or healthcare operation purposes.

consulting physician

Any physician providing a **patient** consult who is not the **primary care physician**. See also: **primary care provider** (PCP).

co-payment

A contractually specified payment required of a health plan member to receive covered services.

Core ETS Terminologies

See: **core terminologies**.

core terminologies

A specific set of **coding scheme**s employed by ETS.

core terminology

A **coding scheme** for which ETS has provided special support, in terms of loaders, APIs, and a predefined ETS coding scheme.

coverage

Services provided under a healthcare plan or policy.

covered entities

Health plans, health care providers, health care clearinghouses, or others who hold or transmit **protected health information** subject to the HIPPA **Privacy Rule** of the **U.S. Department of Health and Human Services**. Such entities must conform to the privacy requirements of the **Health Insurance Portability and Accountability Act** of 1996.

CPT code

Coding convention defined by the **Health Care Financing Administration** (HCFA), now the **Centers for Medicare & Medicaid Services**, to identify medical or psychiatric procedures. Used to determine reimbursement amounts to providers by Medicare carriers. A growing number of managed care and other insurance companies also base their reimbursements to their commercial members using this coding convention.

credentials

That set of documents or document references (academic degrees, certifications, licenses, professional affiliations, employment history, references...) that qualify a **practitioner** to deliver healthcare services. Credentials are issued by **external organization**s.

See Also: **verification**.

cross-referencing

Generally refers to the association of related objects across systems. In message processing, the mapping of externally assigned identifiers to HTB-assigned (internal) identifiers.

Current Procedural Terminology

A classification of procedures used for Medicare reimbursement.

D

data controller

An entity that determines the purposes for which and the manner in which **personal health information** is processed, and which has the legal obligation to protect the privacy of personal health information in its control; may or may not correspond to a legal entity. May be a component of a legal entity or several legal entities may be grouped together to form one data controller.

data integrity

The guarantee that the contents of the message received were not altered from the contents of the original message sent.

decryption

The process of converting the contents of an encrypted message (**ciphertext**) back into its original readable format (**plaintext**).

de-identified health information

Health information that neither identifies nor provides a reasonable basis to identify an individual. Individual health information can be de-identified by the removal of specified identifiers of the individual and of the individual's relatives, household members, and employers, and is adequate only if the covered entity has no actual knowledge that could be used to identify the individual.

dependent

A person covered by another person's health plan or policy; in a payer's policy of insurance, a person other than the subscriber eligible to receive care pursuant to subscriber's contract.

Diagnosis-Related Group

A classification system developed by the U.S. Department of Health and Human Services. Patients with similar ICD--9-CM diagnoses who undergo similar procedures are included in the same Diagnosis Related Group. DRGs are used by the United States Medicare system to reimburse hospitals for their treatment of patients, reimbursing a fixed amount for all patients in a DRG regardless of their lengths of stay. DRGs are also widely used in health data analysis.

discharge

Release of an **inpatient**; termination of an inpatient's residence at a medical facility. Changes **patient** status from **pending discharge** to *discharged*.

Domain Message Information Model

A form of the **Refined Message Information Model** constructed to represent the totality of concepts embodied in the individual RMIMs needed to support the communication requirements of a domain. An HL7 version 3 concept.

E

electronic health record

Patient medical record stored in electronic format.

emergency medical services

Medical transportation (ambulance service).

emergency room

Hospital facility staffed and equipped to receive and treat persons with emergent health conditions (illness, trauma...).

emergency room encounter

An **encounter** that occurs at the emergency room.

encounter

Any contact between **patient** and provider where medical or related services are provided; includes consults. Defined by patient, **practitioner** or **clinician**, location, date, and time. Encounters can be related by either **encounter group**s or **encounter link**s.

encounter class

Classifies **encounter** type as **inpatient**, **outpatient**, field, or emergency.

See Also:

- **ambulatory encounter**
- **emergency room encounter**

encounter group

An association between two or more **encounter**s—belonging to the same **patient** or different patients. A group contains attributes and zero to many encounters. Groups can be created to relate a series of encounters for a particular diagnosis or referral—some times called an episode of care. A group can also be created to relate different patient encounters for a clinical study.

encounter link

An association between two related **encounter**s, belonging to the same patient or different patients. For example, links can be made between mother and newborn encounters. Links can also be used to associate pre-admit testing with an inpatient stay.

encounter list

List of encounters by patient.

Encounter Management

A principal service of Oracle Healthcare that manages the interactions between patients and providers for the purpose of delivering healthcare services.

encrypted person record

A person record that has been encoded to hide the identity of the person associated with the record.

encrypted text

Text that has been encrypted, using an encryption algorithm; the output stream of an encryption process. On its face, it is not readable or decipherable without first being decrypted. Also called **ciphertext**. Encrypted text originates as **plaintext**.

encryption

The process of disguising text (or a message), rendering it unreadable to any but the intended recipient.

enterprise

A healthcare organization that consists of one or more facilities and employs multiple practitioners and other employees to deliver healthcare services; the principal business entity that defines a healthcare organization.

Evaluation and Management Guidelines

Identify separate services or procedures beyond the scope of those normally required by **CPT code**d services or procedures; used for billing purposes. Also called E&M Guidelines.

external organization

Any organization outside of the **enterprise**, such as medical groups, medical practices, payers. Other HTB objects (such as persons) may be affiliated with such external organizations. External organizations issue **credentials**.

F

facility

Physical site of a healthcare organization, where healthcare services are delivered; a healthcare institution, such as a hospital or a clinic.

G

General Timing Specification

Defines the complex timing of events and actions in orders and scheduling systems. GTS supports the cyclical validity patterns that may exist for particular types of information, such as telephone numbers (evening, daytime), addresses, office hours. An HL7 version 3 concept.

Graphic User Interface

An interface used with personal computers and workstations that lets users access window fields and regions with a pointing device, typically a mouse. Also called a **window user interface**, or a GUI.

guarantor

An individual (or other entity) who assumes financial responsibility for the healthcare received by another party (a **patient**). The guarantor guarantees payment for healthcare services provided to the patient.

H

health care clearinghouse

An entity that processes or converts healthcare information from or to a standardized format or data content in the course of its routine processing. Such organizations include billing services, repricing companies, community health management information systems, and value-added networks and switches that perform clearinghouse functions.

Health Care Financing Administration

An agency of the **U.S. Department of Health and Human Services**, HCFA administers Medicare, Medicaid, and SCHIP. The agency also performs a number of quality-focused activities, including regulation of laboratory testing (CLIA) and development of coverage policies. Name changed to **Centers for Medicare & Medicaid Services** as at July 1, 2001.

health care professional

See: **practitioner**.

health care provider

An individual or organization licensed, certified, or otherwise authorized to deliver healthcare services.

See: **practitioner**.

Health Insurance Portability and Accountability Act

The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Title I of the act provides health insurance coverage for workers and their families when they change or lose their jobs. Title II of the act requires the U.S. Department of Health and Human Services (HHS) to establish national standards for electronic healthcare transactions and national identifiers for providers, health plans, and employers. The act also addresses security and privacy of healthcare data.

Health Level Seven

A healthcare application protocol for electronic data exchange. A set of standard formats that specify interfaces between computer applications from different vendors; lets healthcare institutions exchange key sets of data from different applications. Also called HL7.

Health Level Seven, Inc.

One of several ANSI-accredited Standards Developing Organizations (SDOs) operating within the healthcare community; emphasizes clinical and administrative data. Headquartered in Ann Arbor, Michigan, this organization asserts and retains copyright in all works contributed by members and non-members relating to all versions of the Health Level Seven (HL7) standards and related materials. To contact this organization:

Health Level Seven, Inc.
3300 Washtenaw Avenue
Suite 227
Ann Arbor, MI 48104
734.677.7777
734.677.6622 (fax)
hq@hl7.org

health plan

An individual or group plan that provides or pays the cost of healthcare services for its members.

Hierarchical Message Description

The specification of the fields of a message and their grouping, sequence, optionality, and cardinality. Contains message types for one or more interactions, or that represent one or more common message element types. The primary normative structure for HL7 messages.

History of Present Illness

Description of events or symptoms preceding the chief complaint.

Hypertext Markup Language

A markup language used to format documents, predominantly for viewing with a web browser. Portions of text or images, called hypertext, can be associated with other documents. Also called HTML.

Hypertext Transfer Protocol

The TCP/IP-based network protocol used to transmit requests and documents between an HTTP server and a web browser. Also called HTTP.

I

inpatient

A **patient** that has been admitted to a medical facility for treatment; a resident patient (for more than 23 hours).

inpatient encounter

An **encounter** involving an inpatient.

instance

A case or an occurrence. For example, an object is an instance of a class.

instance identifier

Used to uniquely identify an **instance** or **object**.

Integrity

See: **data integrity**.

interterminology equivalence

Semantic equivalence between concepts in different ETS **coding scheme**s. Interterminology Equivalence information is loaded into ETS using cross maps designated as having *equivalence context.*

See Also: **intraterminology equivalence**, **interterminology mapping**

interterminology mapping

Mapping between concepts in versions of different **coding scheme**s. Interterminology mapping is carried out using the ETS Cross Map Model.

See Also: **interterminology equivalence**, **intraterminology equivalence**

intraterminology equivalence

Semantic equivalence between concepts in the same ETS **coding scheme**. Concepts may be implicitly equivalent (the concepts bear the same **concept code** in different versions and the coding scheme maintainer has not declared a code re-use to have occurred), or explicitly equivalent (the concepts bear different concept codes, but the coding scheme maintainer has declared that a reassignment has occurred). Intraterminology equivalence information is loaded into ETS from change files.

See Also: **interterminology equivalence**, **interterminology mapping**

J

Java class

Java Classes are components of a Java program that define objects and operations performed on objects. A Java class also identifies an operating system file that contains a program or part of a program written in Java.

Java Server Pages

Java Server Pages (JSP) are an extension to the Java servlet technology that was developed by Sun Microsystems as an alternative to Microsoft ASPs (Active Server Pages). JSPs support dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements—the actual design and display of the page.

JInitiator

Oracle JInitiator lets end users run Oracle Developer server applications directly within Netscape Navigator or Microsoft Internet Explorer on the Windows 95, 98, 2000, and NT4.0 platforms. Implemented as a plug-in (Netscape Navigator) or

ActiveX component (Microsoft Internet Explorer), Oracle JInitiator lets you specify the use of Oracle's Java Virtual Machine (JVM) on web clients in lieu of the browser's default JVM.

K

L

List of Purposes

List of valid purposes for which the processing of **personal health information** is authorized by law or local regulation. Required by the HIPPA **Privacy Rule**, as well as privacy regulations of several countries and geographic locales.

location

The healthcare facility, clinic, hospital unit, room, and bed; the location of an **encounter**.

M

Major Diagnostic Category

Grouping by admitting diagnosis. Assignment of the MDC is the first step in the process of formulating the **Diagnosis-Related Group**, necessary for reimbursement of a healthcare claim.

master catalog

The organization of **clinical act**s by type. Ties a particular organization's clinical data to Oracle Enterprise Terminology Services (ETS).

medical record number

A reference number that uniquely identifies a **patient** medical record or a patient.

medical record review

Periodic review of the patient's medical record for quality improvement studies, research projects, billing compliance auditing, and claims review. See also: **electronic health record** (EHR).

medication history

Describes medications and dosages prescribed for a patient

membership code

Uniquely identifies a **concept** within a **concept list**.

message

A unit of data transferred between systems or applications.

Message Type

A three-character code imbedded in a **message** that defines the purpose of the message. Example: ADT; indicates that the purpose of the message is to transmit ADT data.

mood

A code specifying whether an act is an activity that has happened, can happen, is happening, is intended to happen, or is requested or demanded to happen. An HL7 version 3 concept.

N

network service

A network resource used by clients; for example, an Oracle database server.

node

See: **organization node**.

O

object

A programming construct that contains both data and procedure to access, update, or use such data by performing a service. An object exists within a class (*an instance of a class*), which defines common characteristics and behaviors for all objects within the class. Objects are abstractions that can represent real-world items, such as a motor vehicle, or a process, such as transferring a **patient**.

object identifier

A globally unique string (example: 2.16.840.1.113883.3.1) that expresses a tree data structure; an ISO concept.

organization node

A group, practice, department, or other unit within an organization. An organization can have several **care site practice setting**s across multiple organization nodes.

outpatient

A patient that has not been admitted to a medical facility for treatment (for more than 23 hours); a non-resident patient; one who visits a hospital, clinic, or associated medical facility for diagnosis or treatment but is not admitted. See also: **inpatient**.

outpatient encounter

An **encounter** involving an outpatient. Examples include visits to the ER, other physician visits or checkups, day surgery, blood donation.

P

participation

An association between a **role** and an act. Represents the involvement of the entity playing the role with regard to the associated act. A single role may participate in multiple acts and a single act may have multiple participating roles. A single participation is always an association between a particular role and a particular act. An HL7 version 3 concept.

patient

A person who receives professional services from a **practitioner** of the healing arts toward the maintenance, improvement, or protection of health, or lessening of illness, disability or pain.

patient authorization

Permission given by a **patient** or individual to use and disclose specific **protected health information** identified for the limited purposes requested. Health information that is used for treatment, payment or health care operations, or is otherwise permitted or required by the **Privacy Rule**, is not protected.

payor

A third-party purchaser of healthcare services. A payor may be an insurance company or a governmental program, such as Medicare or Medicaid.

pending admission

The first step in admitting an **inpatient**, it alerts a designated **practice setting** to the imminent arrival of a new patient and collects required **encounter** data prior to the actual admission; defines the commencement of a new encounter.

pending discharge

The first step in discharging an **inpatient**, it alerts the appropriate ancillary departments to the **patient**'s imminent discharge. This enables them to schedule the termination of services to that patient upon the patient's departure.

performer

A person participating in delivery of a service. A **participation** type code depicting the kind of participation or involvement the actor (the entity playing the **role** associated with the participation) has with regard to the associated act.

personal health information

Data created or received by a **data controller** that relates to (i) the physical or mental health or condition of an individual, or (ii) the provision of healthcare to an individual, or (iii) payment for healthcare services to an individual.

Person Services

A service of Oracle Healthcare. Includes a directory that uniquely identifies persons, who may be patients, providers, payers, other medical staff, or all of these. It can exist within a single domain or an enterprise with multiple domains, and can be employed at the organization, community, state, national, or international level. Previously called Enterprise Master Person Index (EMPI).

PL/SQL

PL/SQL is a procedural extension of SQL that provides programming constructs such as blocks, conditionals, and functions.

plaintext

Message or other text that has not been encrypted.

port

In TCP/IP and UDP networks, a port is an endpoint to a logical connection. The port number identifies what type of port it is. For example, port 80 is used for HTTP traffic.

practice setting

A categorization of the clinical setting (cardiology clinic, primary care clinic, rehabilitation hospital, skilled nursing facility...) in which care is delivered. (Note that there is a many-to-many relationship between practice setting and the physical location where care is delivered. Thus, a particular room can provide the location for cardiology clinic one day, and for primary care clinic another day; and cardiology clinic might be held at one physical location on one day, but at another physical location on another day.) Practice settings are **node**s in an organization hierarchy. Examples of practice settings include *MedSurg*, *ICU3*, or *Physical Therapy*. Practice settings are typically part of larger organizational units.

practitioner

A medical professional or technician licensed or otherwise authorized or permitted by law to provide healthcare services. A practitioner may be a physician, nurse, or other type of licensed healthcare professional. Also called healthcare professional, **clinician**.

pre-admission

An optional process conducted in addition to **pending admission**, this process is conducted when episode related procedures must be performed in preparation for a **patient**'s prospective admission to a healthcare facility. Examples include diagnostic tests, blood typing, radiology studies, x-rays.

pre-registration

A service of Oracle Healthcare that lets a prospective **patient** enter all required information and complete all required **admission** forms prior to admission.

primary care physician

See: primary care provider. Note that these terms are often used interchangeably. Also called PCP.

primary care provider

The principal **practitioner** providing or coordinating the delivery of healthcare services to a particular patient. May also be called primary care physician, if the practitioner is a physician. Also called PCP.

Privacy Rule

Issued by the **U.S. Department of Health and Human Services** pursuant to provisions of HIPPA, the **Health Insurance Portability and Accountability Act**, to establish standards of privacy for individually identifiable health information.

Covers **protected health information** held or transmitted by a covered entity or its business associate, in any form or media. Privacy regulations concerning the access of **personal health information** are implemented as a set of privacy rules enforced by HTB Security Services. Also a security rule that specifies permitted use or disclosure by a **data controller**.

privileges

Provide access to a healthcare institution's resources (personnel, equipment, facilities, care sites...) for the purpose of providing patient care. Privileges are granted on the basis of credentials, and the assessment of a **clinician**'s qualifications to perform associated services.

procedure type

Describes actions to be performed, associated **CPT code**s, and optional resources required.

profile option

A profile option is a set of changeable attributes that affect the way Oracle applications appear and how they function.

protected health information

Personal health information that is maintained or transmitted in any form or medium. HIPAA privacy rules apply only to protected health information—which includes **personal health information** but excludes (i) education records covered by the Family Educational Rights and Privacy Act, and (ii) employment records maintained by a covered entity in its capacity as an employer. Individually identifiable health information is information, including demographic data that relates to:

- The individual's past, present or future physical or mental health or condition.
- The provision of health care to the individual.

and that identifies the individual or for which there is a reasonable basis to believe can be used to identify the individual; includes many common identifiers, such as name, address, birth date, social security number.

- The past, present, or future payment for the provision of health care

provider

An individual or organization that is licensed to deliver medical care. All **clinician**s, hospitals, clinics, and other types of healthcare organizations are providers.

proxy server

An intermediate server positioned between a client application, such as a Web browser, and another target server. It intercepts all requests to the target server to see if it can fulfill the requests itself. If it cannot do so, it forwards the request to the target server.

Q

R

read-only

Read-only access lets you access data without changing it.

receiver

A single HTB enterprise that is addressed by an HL7 version 3 message. Receivers are identified by the root of the Instance Identifier of the `OrganizationRCV` object in the message wrapper of HL7 messages that conform to Oracle Conformance Specifications. Called the `ReceiverIdentifier` in IMP configuration. Must be a valid cross reference to an HTB enterprise.

Reference Information Model

An object model created as part of the HL7 version 3 methodology, RIM is a pictorial representation of the clinical data (domains) and identifies the life cycle of events that a message or groups of related messages will carry.

Refined Message Information Model

An information structure that represents the requirements for a set of messages. A constrained subset of the **Reference Information Model**. May contain additional classes that are cloned from RIM classes. Contains those classes, attributes, associations, and data types that are needed to support one or more **Hierarchical Message Description**s. A single message can be shown as a particular pathway through the classes within an RMIM. An HL7 version 3concept.

registration

A service of Oracle Healthcare that captures information (demographic, financial, other) necessary to establish a new **patient** in the system. The submitted information is received by the medical facility's admitting department and processed daily.

resource

Any person, **care site**, equipment, or other facilities or capabilities required for a scheduled appointment. Example: The resources to support a laser treatment could include a care site and a laser technician.

responsibility

A level of authority within Oracle Applications that lets users access those application functions and data that are consistent with their assigned roles within an organization. Although user accounts can be granted multiple responsibilities within Oracle Applications, more than one responsibility cannot be active at the same time during a user session (users select one responsibility at a time).

role

Within **Health Level Seven**, a role refers to the function or responsibility assumed by a person or organization in the context of healthcare events or activities. For example, a person can have the role of **clinician**, while an organization can have the role of **practice setting**.

Rosetree

An HL7 version 3 tool used to develop **Hierarchical Message Description**s and **Message Type**s from **Refined Message Information Model**s.

Rosetree RIM Browser

An HL7 tool that lets you browse (read) the **Reference Information Model** as a UML model, including RIM data types and vocabulary.

S

sender

An application that is capable of originating HL7 version 3 messages. Senders are identified by the Instance Identifier of the `DeviceSND` object in the message wrapper of HL7 messages that conform to the Oracle Conformance Specifications for such messages; called the `SenderIdentifier` in IMP configuration.

server

A computer that is accessed by and serves other client or server computers in a network. Examples include mail servers, database servers, and applications servers (Oracle9*i*AS).

service

A network resource used by clients; for example, an Oracle database server.

servlet

A servlet is a Java program called or triggered by a client computer, executed on an HTTP server.

side effect

Within the context of HL7 version 3 messaging, the creation, updating, or replacement of an object with information from a message, when that object is not the focus of the message.

SQL

Structured Query Language. An internationally standardized language used to access data in a relational database.

SQL*Plus

An Oracle language superset of SQL used to submit SQL statements to an Oracle database server for execution. SQL*Plus has its own command language.

SQL script

A SQL script is a file containing SQL statements that you run with a tool such as SQL*Plus to query or update an Oracle Relational Database.

substance administration

An act using a material as a therapeutic agent. The effect of the therapeutic substance is typically established on a biochemical basis, but that is not a requirement. An HL7 version 3 concept.

system administrator

The person who manages administrative tasks in Oracle Applications, such as registering new users and defining system printers, or granting the system administrator responsibility to other users.

T

TCP/IP

Transmission Control Protocol/Internet Protocol. A widely used industry-standard networking protocol used for communication among computers. The communication standard of the Internet.

tier

A set of machines that perform similar tasks. Client/server is a two-tier architecture, with machines on the client tier connecting to machines on the server tier. Internet Computing Architecture consists of three tiers. In Oracle Applications Release 11*i*, machines on the desktop client tier communicate with machines on the application tier, which in turn communicate with each other and with machines on the database tier.

transfer

Movement of a **patient** from one **practice setting** to another. May imply change in patient's level of care.

Transmission Control Protocol / Internet Protocol

See: **TCP/IP**.

Trigger Event

Within the context of HL7 version 3 messaging, the `TriggerEvent` is the unique combination of a message type and a state transition that the focal class of the message type can undergo. Each HL7 version 3 message identifies its trigger event using the code attribute of the `ControlActEvent` class in the `ControlAct` wrapper (called the `TriggerEventCode`).

U

Unified Modeling Language

An industry standard tool for object-oriented analysis and design. Used to create domain models, UML was originally created to unify several well known object-oriented modeling methodologies, principally including those of Grady Booch, Jim Rumbaugh, and Ivar Jacobson.

Unified Service Action Model

Describes of the basic structures of the HL7 **Reference Information Model** (RIM); the clinical part of the RIM.

U.S. Department of Health and Human Services

Administers Health Insurance Portability and Accountability Act (HIPAA), among other functions relating to the regulation of the healthcare industry in the United States. Also called HHS.

user

A user is any person requiring access to an application, including various types of customers, partners, suppliers, and employees.

user name

A unique name that grants access to a secure environment or program, such as an Oracle database or Oracle applications. A user name is typically associated with a collection of privileges and data available to a particular user (*responsibilities* in Oracle Applications). User names are normally associated with a password.

V

verification

Within Credentials Management, the act of authenticating **credentials**.

W

window user interface

An interface used with personal computers and workstations that let users access window fields and regions with a pointing device, typically a mouse. Also called a **Graphic User Interface**.

workflow

Oracle Workflow automates business processes, routing information of any type according to uniquely defined business rules. These rules, called a *workflow process definition*, include the activities that occur in the process and the relationship between those activities. An activity in a process definition can be an automated function defined by:

- A PL/SQL stored procedure or an external function.
- A notification to a user or responsibility that they may request a response.a business event
- A subflow that itself is made up of multiple activities.

workflow attributes

Workflow attributes control the behavior of the workflow.

workflow monitor

The workflow monitor is a Java based tool used for administering and viewing workflow processes.

X

Y

Z